10:16:53 OCA PAD INITIATION - PROJECT HEADER INFORMATION 05/12/89

Active

Project #: E-25-M83 Cost share #: E-25-352 Rev #: 0
Center # : 10/24-6-R6737-0A0 Center shr #: 10/221-F6737-0A0 OCA file #:
Work type : RES
Contract#: ECS-8909251 Mod #: Document : GRANT
Prime #: Contract entity: GTRC

Subprojects ? : N
Main project #:

Project unit: ME Unit code: 02.010.126
Project director(s):
RUSHMEIER H E ME (404)894-3208

Sponsor/division names: NATL SCIENCE FOUNDATION / GENERAL
Sponsor/division codes: 107 / 000

Award period: 890915 to 920229 (performance) 920528 (reports)

Sponsor amount	New this change	Total to date
Contract value	60,000.00	60,000.00
Funded	60,000.00	60,000.00
Cost sharing amount		15,000.00

Does subcontracting plan apply ?: N

Title: PROGRESSIVE REFINEMENT ALGORITHMS FOR RADIANT TRANSFER

PROJECT ADMINISTRATION DATA

OCA contact: David B. Bridges 894-4820

Sponsor technical contact

GEORGE LEA
(202)357-9618
NATIONAL SCIENCE FOUNDATION
ENG/ECS
WASHINGTON, D.C. 20550

Sponsor issuing office

SHIRLEY A. WOODS
(202)357-9602
NATIONAL SCIENCE FOUNDATION
DGC/ENG
WASHINGTON, D.C. 20550

Security class (U,C,S,TS) : U
Defense priority rating : N/A
Equipment title vests with: Sponsor

ONR resident rep. is ACO (Y/N): N
NSF supplemental sheet
GIT X

Administrative comments -
PROJECT INITIATION, NEW FUNDS IAO $60000 W/ COST SHARE OF $15000 FROM M.E.
2-YEAR GRANT UNDER NSF "RESEARCH INITIATION AWARDS - 1989" PGM.

3

GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION

NOTICE OF PROJECT CLOSEOUT

Closeout Notice Date 02/17/92

Project No. E-25-M83 Center No. 10/24-6-R6737-0A0

Project Director RUSHMEIER H E School/Lab MECH ENGR

Sponsor NATL SCIENCE FOUNDATION/GENERAL

Contract/Grant No. ECS-8909251 Contract Entity GTRC

Prime Contract No.

Title PROGRESSIVE REFINEMENT ALGORITHMS FOR RADIANT TRANSFER

Effective Completion Date 920229 (Performance) 920528 (Reports)

Closeout Actions Required:	Y/N	Date Submitted
Final Invoice or Copy of Final Invoice	N	
Final Report of Inventions and/or Subcontracts	Y	920204
Government Property Inventory & Related Certificate	N	
Classified Material Certificate	N	
Release and Assignment	N	
Other	N	

Comments98A SUBMITTED. NO INVOICE--LETTER-OF-CREDIT.

Subproject Under Main Project No.

Continues Project No.

Distribution Required:

Project Director	Y
Administrative Network Representative	Y
GTRI Accounting/Grants and Contracts	Y
Procurement/Supply Services	Y
Research Property Managment	Y
Research Security Services	N
Reports Coordinator (OCA)	Y
GTRC	Y
Project File	Y
Other	N
	N

NOTE: Final Patent Questionnaire sent to PDPI.

Annual Progress Report

for

Grant ECS-8909251

"Progressive Refinement Algorithms for Radiant Transfer"

submitted by:

Holly E. Rushmeier
Assistant Professor
The George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology
Atlanta, GA

November 27, 1990

Annual Progress Report for
Grant ECS-8909251
"Progressive Refinement Algorithms for Radiant Transfer"

1. Summary of Overall Progress

The goal of this research is the development of progressive refinement algorithms for radiant energy transfer. In the original proposal this work was to be performed in two phases corresponding to the two years of the grant. In phase I, two lines of research were to be pursued: the development of hierarchical geometric and property descriptions, and the development of incremental solution systems. In phase II the hierachical descriptions were to be integrated into the incremental solution systems.

Of the two lines of reseach in phase I, the greatest success has been achieved in the development of incremental solution systems, since this is a continuation of research which was initiated at the time the original proposal was submitted. The first "layer" of an incremental solution is a progressive refinement radiosity solution. New results have been obtained to two areas of progressive refinement radiosity solutions -- the calculation of form factors, and the calculation of non-diffuse interreflections.

In a project involving Mechanical Engineering graduate student David Hall, and Dan Baum from Silicon Graphics, Inc., we investigated the use of computer graphics hardware for accelerating form factor calculations for a progressive refinement radiosity solution. We found the fundamental limitation of the possible speedup obtained by using the graphics hardware, and developed algorithms to speed up the remaining bottlenecks in the form factor calculations. This work was presented at the AIAA/ASME Joint Heat Transfer Conference in Seattle, WA, in June, 1990. A copy of the paper is appended to this report. The paper has subsequently been submitted to the *Journal of Heat Transfer*.

To have an accurate incremental solution, a radiosity solution used for a preprocess to a Monte Carlo solution must include the effects of non-diffuse interreflections. In the research for his Master's Thesis, David Hall examined the method for non-diffuse reflections proposed by Shao, Peng, and Liang in the proceedings of SIGGRAPH 1988. After presenting a corrected formulation of the method, he presented a progressive refinement implementation of the method, examined an adaptive refinement approach for discretizing non-diffuse surfaces, and developed a more efficient method for performing the integration of the product of bidirectional reflectance times intensity over the incident hemisphere for surfaces which are neither Lambertian nor mirrorlike. The thesis summary page describing these results is appended to this report. Hall's Master's thesis was given an Outstanding Master's Thesis Award by the Georgia Tech chapter of Sigma Xi. A journal paper describing Hall's work is currently in preparation.

One problem in developing progressive refinement solutions is determining the accuracy required in the final solution. In the case of illumination calculations for computer graphics animation, the solution for any particular geometry will only visible for 1/30 of a second (i.e. for one frame of the animation). If each frame were a highly accurate radiosity solution, the environment would have to be remeshed for each frame. Charles Patterson, a Ph.D. student in the College of Computing, Brian Guenter a faculty

member in the College of Computing, and I worked on developing various temporal filters for an animation to reduce the spatial sampling required for radiosity solutions. A short note describing these results is in preparation.

Much less progress has been made in the development of hierarchical property and geometric descriptions. Most of the work in this area has been in evaluating reflectance models. My work has concentrated on looking for models for which it is straightforward to fit experimental data, rather than models based on surface microgeometry. Two promising models are a model developed by Ward at Lawrence Berkeley Lighting Laboraties, and a model for infrared signature work developed by Sandford and Robertson. Besides comparing which of these models would be best to use in an incremental solution system, an issue which I hadn't anticipated is how to evaluate the fit to experimental data. One approach is a simple minimization of rms error data point by data point. Another approach is to optimize the fit to various features of the experimental data -- such as the relative height and width of the lobe of maximum reflection. More experimental data is required to resolve this issue.

2. Current Problems and Favorable or Unusual Developments

The major problem in performing this research is in the area of graduate student support. I have not yet had a student eligible for cost sharing, and nearly the entire amount budgeted for two graduate students has been spent to support one full time graduate student. From Sept. 1989 to March 1990, this student was David Hall, a Master's student in the School of Mechanical Engineering. From April 1990 to the present, this student has been Charlie Patterson, a doctoral student in the College of Computing. Research progress has suffered some from not having overlap between these students.

The development of a hierarchical system which employs various levels of geometric description is quite ambitious at the current rate of progress. To alleviate this problem, a small grant of funds and equipment has been obtained from the Apple Computer Company specifically to examine the use of a hierarchy of geometric descriptions in the generation of computer graphics images.

The project described in section 1 above regarding temporal filtering of radiosity results has opened a new line of research which wasn't anticipated in the original proposal. As well as developing progressive refinement solutions for static geometries, progressive refinement solutions can also be generated for animated sequences. Nearly all of the various techniques for spatial sampling of static geometries can be extended to the development of techniques for temporal sampling.

3. Summary of Work to be Performed in the Next Budget Period

There are two main projects to be performed in the next year:

The first is the implementation of a progressive refinement system using a hierarchical description of the environment geometry. In the initial implementation, the crude and detailed geometric descriptions of each object will be defined manually. A progressive refinement radiosity solution will be performed on the crude description. A Monte Carlo ray tracing solution will then be performed on the detailed description,

with the radiosity solution used for secondary reflections. This implementation will be performed by Charlie Patterson, and by a second student to be supported by the grant from Apple mentioned in section 2.

Second, the application of progressive refinement methods to heat transfer problems needs to be explored. The results of Hall's revision of the method developed by Shao et al. will be compared to solutions published for non-diffuse surfaces in the heat transfer literature. Since most published heat transfer solutions are for extremely simple geometries, some methods from heat transfer will have to be implemented to perform more extensive comparisons.

Annual Progress Report for ECS -8909251
Attachment #1
Paper presented at AIAA/ASME Thermophysics and Heat Transfer Conference, held June 18-20, 1990, Seattle, WA.
A revised version of the paper has been submitted to The Journal of Heat Transfer.

ACCELERATING THE HEMI-CUBE ALGORITHM FOR CALCULATING RADIATION FORM FACTORS

H. E. Rushmeier
George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology
Atlanta, Georgia

D. R. Baum
Silicon Graphics Computer Systems
Mountain View, California

D. E. Hall
George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology
Atlanta, Georgia

ABSTRACT

Determining radiation form factors when shadowing effects are considered is computationally expensive. For a system of N surfaces, brute force methods require $O(N^3)$ time to compute the required N^2 factors. In computer graphics, Cohen introduced the hemi-cube algorithm which calculates the N^2 factors in $O(N^2)$ time. In this paper, methods which speed up the original hemi-cube algorithm by taking advantage of graphics hardware and two types of geometric coherence are presented. The implementation of the methods on a computer workstation is described, and timings demonstrating speedups by factors of up to 6.7 are given.

INTRODUCTION

The calculation of geometric form factors for the analysis of radiation heat transfer is complex when surfaces are not in full view of one another. For an enclosure of N surfaces, a total of N^2 factors is needed. When some surfaces shadow one another, brute force methods require $O(N^3)$ time to compute the required factors (Walton, 1987). Recently Cohen's hemi-cube algorithm (Cohen and Greenberg, 1985) which calculates the N^2 factors in $O(N^2)$ time, has become popular in computer graphics. The hemi-cube algorithm simplifies form factor calculation by allowing the user to define only the geometry of the objects in the enclosure, without the need for visibility information and special cases. Because the algorithm is fast and straightforward, methods to further improve the hemi-cube algorithm by taking advantage of graphics hardware and two types of geometric coherence will be presented.

The acceleration of the calculation of form factors is not an isolated computational problem. The ability to calculate form factors very quickly can alter the overall computational methodology used to analyze radiant heat transfer, and can greatly extend the complexity of geometries which can be studied.

REVIEW OF THE HEMI-CUBE ALGORITHM

Background

The hemi-cube algorithm was developed in computer graphics to perform the illumination calculations necessary to generate realistic synthetic images. The relationship between methods in computer graphics and radiant heat transfer have been recognized by many researches (e.g. Eichberger, 1985, Emery and Abrous, 1987 and Howell, 1989). Specifically, hidden surface methods from computer graphics have been used by Emery et al. (1987) and Walton (1987) to calculate form factors. In these methods however, form factors are computed one pair at a time, resulting in algorithms of complexity $O(N^3)$. In the hemi-cube algorithm, the form factors from a surface A_i to all other surfaces A_j are calculated *simultaneously*. Calculating the factors simultaneously eliminates determining redundant visibility information and results in an algorithm of complexity $O(N^2)$.

To present the new acceleration techniques for the hemi-cube algorithm, a description of the original algorithm is needed. A general description will be given. The details of the computer graphics techniques involved can be found in standard texts such as Foley and van Dam (1982) and Newman and Sproull (1979).

Description of the Original Hemi-Cube Algorithm

The form factor F_{ij} from A_i to A_j is given by the integral:

$$F_{ij} = (1/A_i)\int_{Ai}\int_{Aj} \cos\theta_i \cos\theta_j \, dA_i \, dA_j/\pi r_{ij}^2$$

where A_i and A_j are the areas of the two surfaces, the angles θ_i and θ_j are measured from the surface normals and r_{ij} is the distance between the dA_i and dA_j, as shown in Fig. 1. The hemi-cube algorithm approximates the form factor F_{ij} from an area A_i to area A_j by the factor F_{dij} from a differential area dA_i at the

center of area A_j. The factor F_{ij} is approximated by:

$$F_{ij} = \int_{Aj} \cos\theta_i \cos\theta_j \, dA_j / \pi r_{ij}^2$$

Surfaces in the enclosure are subdivided into small subsurfaces to make this approximation valid.

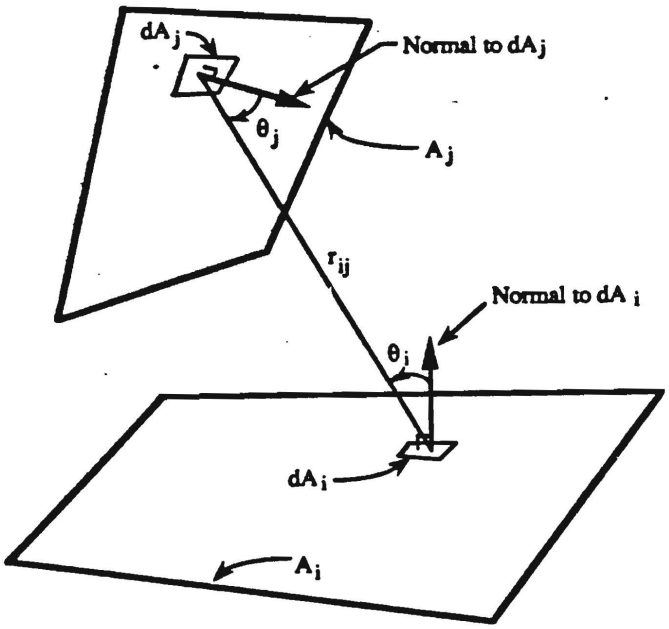


Figure 1: Form Factor Geometry

Similar to other numerical methods for finding form factors (such as those described by Maxwell et al., 1986 and O'Neill and Zich, 1985) the hemi-cube method is based on Nusselt's analogy, which is illustrated in Fig. 2a. It is clear from Nusselt's analogy that from a differential area dA_i, any two surfaces which project onto the same portion of the unit hemisphere have the same form factor. To use this idea in a numerical method, the hemisphere is discretized into many small surfaces as shown in Fig. 2b. Form factors from dA_i to the surfaces on the discretized hemisphere can be found analytically. Form factors to the real surfaces in the environment can be approximated by finding the closest real surface visible through each of the small surfaces on the hemisphere. The sum of the form factors associated with the small surfaces on the hemisphere through which a particular real surface is seen is equal to the form factor of the real surface. The hemisphere must be discretized into a large number of small surfaces to calculate accurate form factors.

Finding form factors using a discretized hemisphere requires an efficient method of determining visibility. The brute force approach of calculating the distance to each of N surfaces in the enclosure from dA_i through each of n small surfaces on the hemisphere would require Nn intersections of surfaces and rays. Using computer graphics algorithms, the number of comparisons can be reduced to Nm, where m is the average number of hemisphere surfaces onto which each real surface A_j

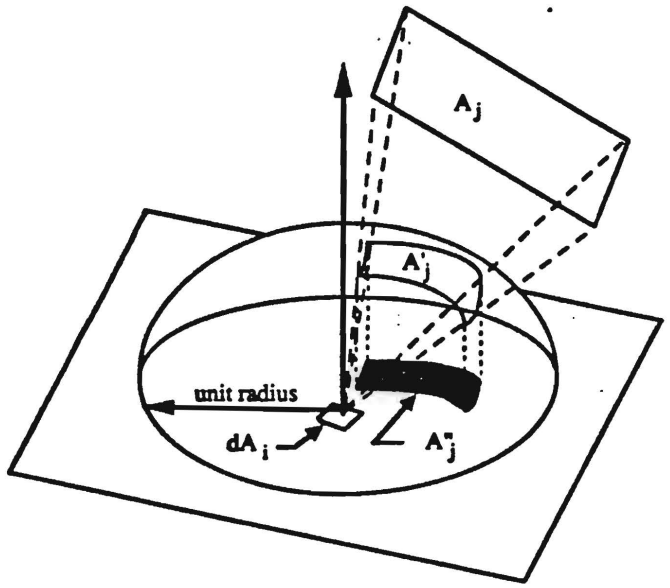


Figure 2a: Nusselt's analogy – The projection of surface A_j onto a unit hemisphere surrounding dA_i is A_j'; the projection of A_j' onto a unit circle around dA_i is $A_j"$. The form factor F_{ij} is equal to the ratio of $A_j"$ to the area of the unit circle.

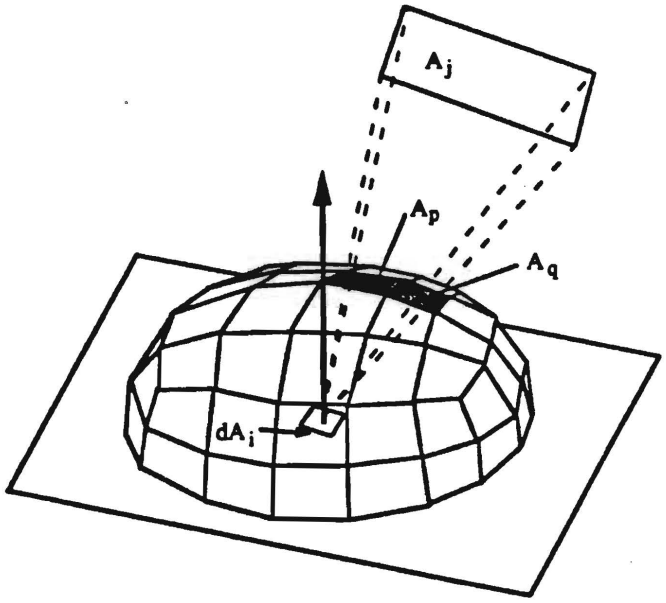


Figure 2b: Using a discretized hemisphere – Surface A_j is visible through the centers of surfaces A_p and A_q, therefore F_{dij} is approximated as $F_{dip} + F_{diq}$.

is projected. Furthermore, the intersection of a ray and plane is not necessary at each comparison. The process of calculating the distance from the eye to each surface can be streamlined using appropriate transformations.

In a computer graphics hidden surface problem, an eye point and screen position are designated. The surfaces in a scene visible at each pixel determine the color to be displayed at that pixel. The method of finding form factors by discretizing the hemisphere above dA_i can be put into the form of a computer graphics hidden surface problem by replacing the hemisphere with a hemi-cube, as shown in Figure 3. The surface dA_i is at the eye position. Each of the five surrounding planes is a "screen". The "pixels" on each screen represent the small fictitious surfaces into which the hemisphere is divided. The frustum of vision is 90 degrees for the top of the hemisphere, and 45 degrees for each of the sides. Since there are five screens, the hidden surface problem is solved five times to determine the form factors from one surface.

The formulation of the hemi-cube algorithm requires that all surfaces in the enclosure be planar polygons. Each surface is visible from only one side, with the visible side indicated by the direction of the normal associated with the surface. Note that this is not a fundamental limitation of the algorithm since any geometry can be approximated to any desired level of accuracy by a set of one-sided planar polygons.

For each surface i in the enclosure, the process begins by transforming the coordinates of all surfaces to a coordinate system centered on dA_i with the positive z axis perpendicular to one of the screens s. After transformation, the surfaces which will not be visible from dA_i through screen s are eliminated in a series of increasingly computationally expensive steps. First, back face elimination is used to exclude from consideration any surface which does not surface i. Back face elimination can be accomplished by examining the transformed value of the surface normal.

Next, the remaining surfaces are clipped against the viewing frustum for screen s. That is, surfaces which do not lie in the frustum formed by the eye and the screen are eliminated, and surfaces which lie partially in the frustum are cut down to just the portion inside the frustum. Clipping can be done efficiently using the Sutherland-Hodgeman clipping algorithm.

The surfaces remaining after clipping are converted to a screen coordinate system using the perspective transformation and a scaling factor. A perspective coordinate system allows the surface in the enclosure which is closest to the eye to be found by a simple comparison of z coordinates. The x and y coordinates are scaled so that they each range from zero to the screen resolution.

The surface visible through each pixel is found using the depth buffer algorithm. Two arrays are needed -- a depth buffer which holds a depth (or z) value for each pixel, and an item buffer which holds a surface identifier for each pixel. Each entry in the depth buffer is initialized to a z value greater than the largest possible z value for any surfaces, and each entry in the item buffer is initialized to a null value. The pixels on the screen covered by each surface j are then identified by finding the pixels onto which the vertices of the polygon will fall and then checking all pixels in the region bounded by the edges connecting the vertices. Because the polygons are planar, the z

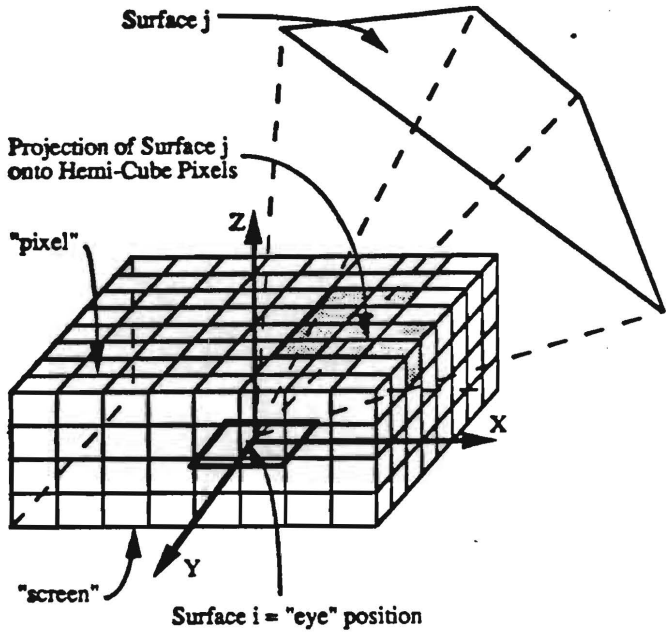


Figure 3: The calculation of F_{ij} using a low resolution hemi-cube.

coordinates at the covered pixels can be found by simple additions to the z coordinate of one vertex. The depth buffer is updated so that it always contains the z coordinate of the closest surface viewed through a particular pixel. The item buffer is updated so that it always contains an identifier corresponding to the closest surface.

After the hidden surface problem has been solved, the form factors from surface i are calculated by scanning the item buffer and summing the form factors associated with the pixels through which a particular surface j is visible. The form factor associated with each pixel is generally referred to as a delta-form factor. The same hemi-cube can be used for all surfaces in the enclosure, so the delta-form factors only need to be computed once for an enclosure.

The hemi-cube algorithm is summarized in the following pseudocode:

```
FOR each hemi-cube pixel q
    Calculate a DeltaFactor(q) equal to the form factor
        to each hemi-cube pixel;
FOR each surface i
    FOR each hemi-cube screen s
        FOR each surface j
            FormFactor(ij) = 0;
            Transform coordinates to system centered on i
                and oriented towards s;
            IF (not back facing)
                Clip to viewing frustum;
                IF(portion of j inside the frustum)
                        Find pixels p onto which j is
                            projected;
                        FOR each pixel p
                            IF (depth of j < depth(p))
                                item (p) = j;
                                depth (p) = depth of j;
                            END IF
                        END FOR each pixel
                END IF portion left
            END IF not back facing
        END FOR each surface j
    END FOR each hemi-cube screen
    FOR each pixel q on the hemi-cube
        FormFactor(i,item(q)) + = DeltaFactor(q);
END FOR each surface i.
```

Accuracy and Performance

The accuracy of the hemi-cube algorithm is discussed in more detail elsewhere (Baum et al., 1989) The accuracy of the method improves with the discretization of the enclosure into smaller surfaces, and the discretization of the hemi-cube into a larger number of pixels. The computation time to calculate form factors increases linearly with the number of pixels on the hemi-cube.

ACCELERATION OF THE HEMI-CUBE METHOD USING HARDWARE

Description of Hardware Implementation

Recently computer graphics workstations have been introduced which perform the hidden surface problem in hardware, to allow the real time dynamic display of complex objects. Such workstations are now available from a variety of vendors. Although formerly very specialized and high priced, graphics workstations are becoming increasingly affordable (on the order of $20,000) and accessible to a wider range of users. Many finite element analysis packages are being implemented on graphics workstations, to take advantage of the graphics hardware in visualizing the results of lengthy calculations on complex geometries. Consequently, workstations with graphics display hardware are becoming increasingly common in engineering environments.

The specialized graphics processor in a workstation performs most of the operations described above in hardware rather than software. A library of subroutine calls to the graphics hardware is provided with the workstation, and the specific form of the subroutine calls will depend on the make of workstation.

In general, the use of the graphics hardware in the hemi-cube algorithm proceeds as follows. A portion of the graphics screen is allocated to represent a screen on the hemi-cube. If the size of the hemi-cube screen is PxP pixels, this allocation is accomplished by opening a graphics window of size PxP on the graphics screen. The surfaces are then displayed in this window by sending the coordinates of the vertices of each surface to the graphics processor. The integer identifier of each surface is sent to the graphics processor as the color to be displayed. After all the surfaces have been displayed, the item buffer containing the visible surfaces at each pixel is filled simply by reading the color being displayed in the graphics window at each pixel.

The pseudocode is revised as follows when the graphics hardware is employed:

```
FOR each hemi-cube pixel q
    Calculate a DeltaFactor(q) equal to the form factor
        to each hemi-cube pixel;
Allocate a portion of the workstation graphics
    screen to be used as the hemi-cube screen;
FOR each surface i
    FOR each hemi-cube screen s
        FOR each surface j
            FormFactor(ij) = 0;
            Send a list of vertices for surface j to the
                graphics processor;
        END for each surface j
    END FOR each hemi-cube screen
    Read the contents of the workstation screen
        color buffer into the item buffer;
    FOR each pixel q on the hemi-cube
        FormFactor(i,item(q)) + = DeltaFactor(q);
END FOR each surface i.
```

Performance of Hardware Implementation

Typical timing results for the software and hardware implementations of the hemi-cube algorithm are shown in Table 1. All timings are for a Silicon Graphics Personal Iris W-4D20G workstation. The timings include only the time to compute form factors, not the time to write the factors to a file, or to perform radiant transfer calculations using the factors.

Results are shown for two geometries. Geometry 1 is shown in Figure 4a, and consists of a cubical enclosure 1 unit on each side which contains a three dimensional array of 64 cubes, each 0.1 unit on a side, uniformly distributed inside the large cubical enclosure. Geometry 2 is the same cubical enclosure containing two cylinders. The first cylinder has radius 0.1 and height 0.3. Its center is located at (0.7,0.5,0.5), and it has been tilted by rotating it 20 degrees about its local x-axis. The second cylinder has radius 0.2, and height 0.4. Its center is located at (0.3,0.4,0.6), and it has been tilted by rotating it 30 degrees about its local z-axis. Each cylinder is discretized into 140 planar polygons. For each geometry, timings are given for the calculation of all required form factors. Results are given for two different levels of hemi-cube discretization.

The speedup depends on the particular geometry, the number of surfaces and the resolution of the hemi-cube. In

Table 1 the speedup obtained using the hardware ranges from 1.6 for Geometry #1 using a low resolution hemi-cube to 3.5 for Geometry #1 using a higher resolution hemi-cube.

At first, it appears that the speedups are much smaller than would be expected from performing calculations in hardware. Once the vertices of a surface have been sent to the graphics processor, the operations of back face elimination, clipping and updating the depth buffer are sped up by a factor of 100 or more. However, since parts of the algorithm are still performed in software, the overall speed up obtained by using hardware is much lower. The program still needs to feed all of the polygon vertices to the graphics processor, and the addition of the delta-form factors is still performed in software. In the hardware implementation of the hemi-cube algorithm, the operations of sending the vertices to the graphics processor and adding up the delta form-factors account for essentially all of the time to compute factors. The time to perform these two operations depends respectively on the number of polygons and the number of hemi-cube pixels.

TABLE 1

Timings for Form Factor Calculations
on a SGI Personal Iris Workstation

(in CPU second)

Geometry	Hemi-cube Res., R*	Method**	Time	Speedup***
1	50	SW	118	-
		HW	72	1.6
		HWSC	60	2.0
		HWSCPC	54	2.2
1	300	SW	1353	-
		HW	387	3.5
		HWSC	380	3.6
		HWSCPC	201	6.7
2	50	SW	66	-
		HW	42	1.6
		HWSC	35	1.9
		HWSCPC	32	2.1
2	300	SW	855	-
		HW	282	3.0
		HWSC	276	3.1
		HWSCPC	141	6.1

*Number of Hemi-Cube Pixels = $R^2 + 4(.5)(.6R \times .6R)$
**SW = Software Implementation
HW = Hardware Implementation
HWSC = Hardware Implementation with Spatial Coherence
HWSCPC = Hardware Implementation with Spatial Coherence and Pixel Coherence
***Speedup = (Time)/(Time for Software Implementation)

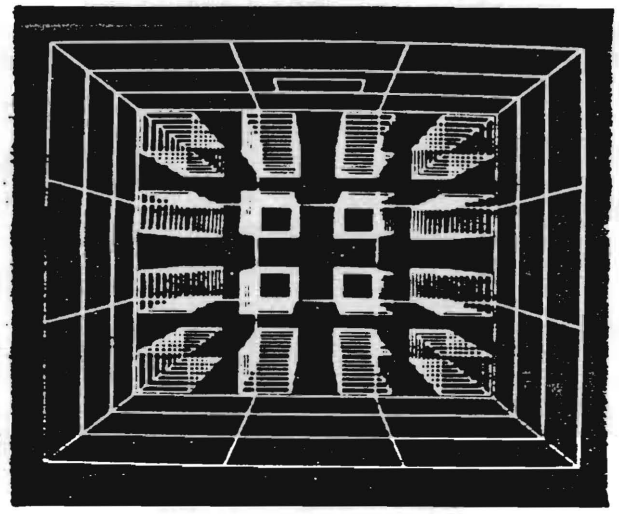


Figure 4a: Test geometry 1 -- Sixty-four cubes evenly distributed in a cubical enclosure.

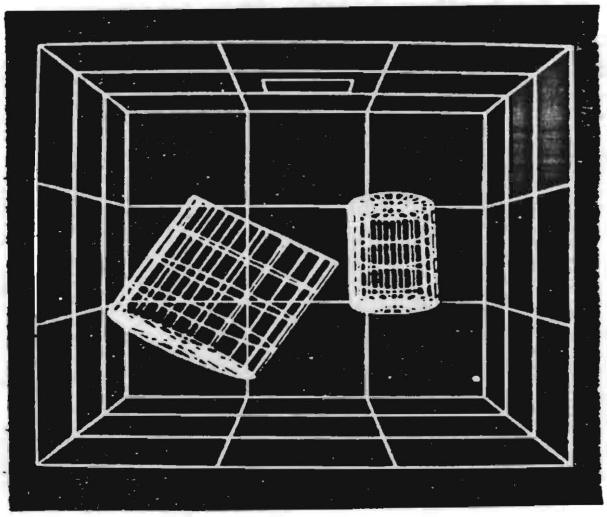


Figure 4b: Test geometry 2 – Two cylinders of different size and orientation in a cubical enclosure. (Note – the use of an aspect ratio less than unity for display resulted in this image of the enclosure appearing to be wider than it is high.)

ACCELERATING THE HEMI-CUBE METHOD USING SPATIAL COHERENCE

Description of Spatial Coherence Algorithm

One step in the process which slows down the calculation of form factors is sending the vertices of each polygon to the graphics processor. One way to accelerate this process is to take advantage of spatial coherence. Groups of polygons which lie behind the current polygon can be identified, so that individual polygons in these groups do not have to be processed. Polygons

can be divided into groups by sorting them into a fixed spatial grid, as diagrammed in two dimensions in Figure 5. Polygon vertices are sent to the graphics processor only if they lie in grid elements which lie in front of the current polygon.

The sorting of polygons into a spatial grid is similar to the spatial subdivision techniques used to accelerate ray tracing in computer graphics (Fujimoto et al., 1984, Glassner, 1986). The idea of sorting surfaces into a spatial grid has been used by Emery et al. (1987) to eliminate potential obstructors between a pair of surfaces. The size of the spatial grid can be set by finding the minimum and maximum x,y,z coordinates in the enclosure. The number of subdivisions in the grid depends on the specific enclosure and generally needs to be supplied by the user based on past experience.

The spatial grid volume into which each surface falls can be determined quickly by comparing the minimum and maximum x,y,z coordinates of the polygon with the coordinates of the spatial grid volumes. The only other operation required is the determination of whether all or part of a grid volume lies in front of a surface. This can be determined by examining the signs of the dot products of the surface normal to each of the vectors from the surface center to the grid volume vertices. If any of the dot products is positive, the grid volume is a least partially in front of the surface.

Specifically, the pseudocode for this variation of the algorithm becomes:

```
Specify the size and discretization of a 3-D grid of
    volumes encompassing the entire enclosure;
FOR each surface i
    Determine which volumes in the spatial grid contain
        surface i or part of surface i;
    FOR each volume v which contains i
        Add i to list of surfaces associated with v;
END FOR surface i
Allocate a portion of the workstation graphics screen
    to be used as the hemi-cube screen;
FOR each surface i
    FOR each surface j
        Front(j) = 0
    FOR each volume v
        IF (v or part of v is in front of i)
            FOR each surface k on the list associated with v
                    Front(k) = 1;
    END FOR each volume;
    FOR each hemi-cube screen s
        FOR each surface j
            FormFactor(i,j) = 0;
            IF (Front(j) == 1)
                    Send a list of vertices for surface j to the
                            graphics processor;
        END FOR each surface j
    END FOR each hemi-cube screen
    (continue as before)
END FOR each surface i.
```

Performance of the Spatial Coherence Algorithm

Typical timing results for the spatial coherence variation of the hemi-cube algorithm are given in Table 1. The overall

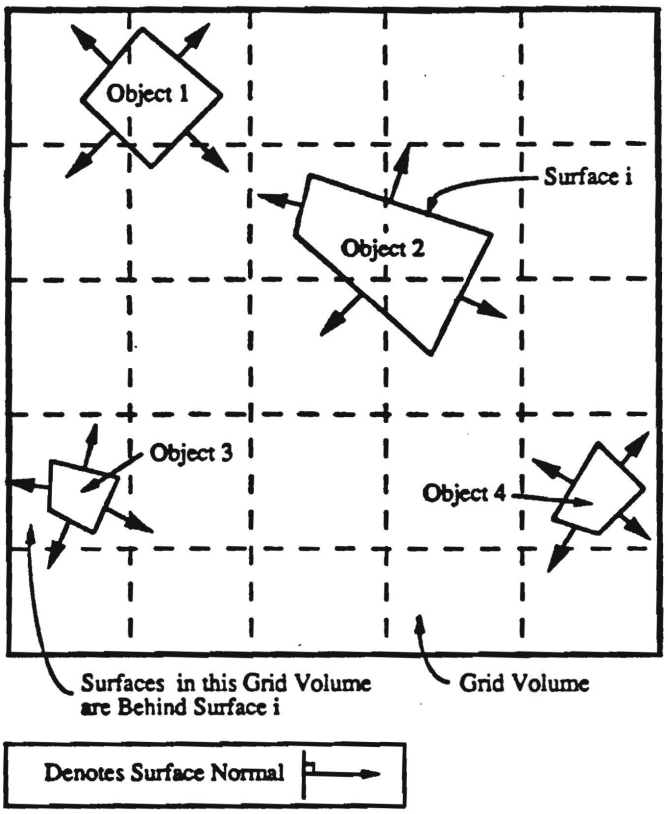


Figure 5: Spatial subdivision in two dimensions -- When computing form factors from surface i, surfaces need to be examined for visibility only if they are contained in grid cells lying at least partially in front of surface i.

speedup over the original software implementation of the hardware version with spatial coherence ranges from a factor of 1.9 to a factor of 3.6 for the results in the table.

Since the spatial coherence algorithm only speeds up the process of sending vertices to the graphics processor, the overall speedup is clearly limited by the fraction of total computational time that is spent sending vertices to the graphics processor. For the higher hemi-cube resolutions in Table 1, the calculations are dominated by the summing of the delta-form factors, and the additional speedup obtained using spatial coherence is very small.

Greater speedups could be obtained by determining which grid volumes will be visible through each of the five screens, and further limiting the number of vertices which have to be sent to the graphics processor. The average number of vertices sent to the graphics processor for each screen could be reduced by as much as a factor of 5. The overall speedup of the form factor calculation produced by this improvement would still depend on the fraction of the overall computation time spent sending vertices to the processor.

In the results given in Table 1, polygons have been treated individually, and each time a polygon is processed all of its vertices need to be sent to the graphics processor. Some

workstations have the option of sending vertices which are shared by multiple polygons to the graphics processor only once, resulting in further speedups.

ACCELERATING THE HEMI-CUBE METHOD USING PIXEL COHERENCE

Description of Pixel Coherence Variation

In general, surfaces will tend to cover contiguous groups of pixels. The summing up of delta-form factors can be sped up by taking advantage of this coherence by updating the value of the form factor for continuous runs of hemi-cube pixels, rather than for every pixel. In this variation, instead of storing a delta-form factor for each pixel, the cumulative form factor is stored for each pixel, i.e. CumulativeFactor(q) is equal to the sum of all DeltaFactor(q') factors for pixels 1 to q. In the loop to add up delta-form factors, an addition is performed only when the value in the item buffer is different for two neighboring pixels.

Pseudo-code for the final portion of the hemi-cube algorithm with this variation is:

```
Start = item(1);
OldValue = 0.;
FOR each pixel q on the hemi-cube
    IF (item(q) not equal to Start):
        TempValue = CumulativeFactor(q-1);
        FormFactor(i,Start) + = TempValue-OldValue;
        Start = item(q);
        OldValue = TempValue;
    END IF item(q)
END FOR each pixel.
```

Performance of the Pixel Coherence Variation

Typical results for the pixel coherence variation (used in combination with the spatial coherence variation) are shown in Table 1. The overall speedup over the original software implementation of the hardware version with spatial and pixel coherence ranges from 2.1 to 6.7. The speedup depends on the average number of pixels through which each polygon is viewed, and on the time required by the computer to perform an integer comparison relative to a floating point addition. Accuracy requires that each surface be visible through substantially more than one pixel, and so this variation will always result in some measurable speedup. As comparison of the low hemi-cube resolution and high hemi-cube resolution entries in Table 1 show, if the number of surfaces is held constant, the speedup obtained by using pixel coherence increases with hemi-cube resolution.

SUMMARY AND CONCLUSIONS

Three methods to accelerate the hemi-cube method for finding form factors have been presented -- use of graphics hardware, use of spatial coherence and use of pixel coherence. The overall speedup resulting from these methods depends on the particular enclosure studied and the required accuracy. Results have been presented illustrating speedups of factors up to 6.7

A number of other methods can be employed to accelerate the hemi-cube algorithm. For example, parallel processing can be employed in the calculations, or alternative strategies for sampling the hemi-cube pixels can be used.

Improving the hemi-cube algorithm is not an isolated computational problem. The ability to calculate form factors very quickly can affect the general procedure used to analyze radiant heat transfer. For example, consider a conventional procedure in which a program is called to compute form factors, requiring $O(N^3)$ time to account for shadowing, and $O(N^2)$ space to store the results. After the form factors are calculated, the conventional method would call a matrix inversion program requiring $O(N^3)$ time and $O(N^2)$ space.

An alternative procedure made possible by fast computation of form factors would be to calculate the form factors for one surface at a time, and use these factors in the Gauss-Seidel iterative method to solve the set of simultaneous equations one row at a time. The Gauss-Seidel method requires $O(N^2)$ time, and the storage required for form factors which are recomputed in each iteration would be only O(N), thus greatly expanding the size of problems which could be studied. Furthermore, rather than waiting for the entire solution to complete before examining the solution, the solution can be displayed on the same graphics display terminal used to calculate form factors. For the initial evaluation of a variety of design alternatives, the solutions could be stopped before convergence to a detailed accurate solution. In this manner many different geometries could be analyzed in a relatively short amount of time.

ACKNOWLEDGEMENTS

This work was funded, in part, by NSF grant ECS-8909251, "Progressive Refinement Algorithms for Radiant Transfer".

REFERENCES

Baum, D. R., Rushmeier, H. E., and Winget, J. M., 1989, "Improving Radiosity Solutions Through the Use of Analytical Determined Form-Factors," *ACM Computer Graphics (Proceedings of SIGGRAPH 89)*, Vol. 23, No. 3, pp. 325-334.

Cohen, M. F. and Greenberg, D. P., 1985, "The Hemi-Cube: A Radiosity Solution for Complex Environments," *ACD Computer Graphics (Proceedings of SIGGRAPH 1985)*, Vol. 19, No. 3, pp. 31-40.

Eichberger, J. I., 1985, "Calculation of Geometric Configuration Factors in an Enclosure Whose Boundary is Given by an Arbitrary Polygon in the Plane," *Waerme und Stoffuebertragung*, Vol. 19, pp. 269-271.

Emery, A. F. and Abrous, A., 1987, "Effects of Specularly Reflected Radiation on Spacecraft Temperatures and Thermal Gradients," *Journal of Spacecraft and Rockets*, Vol. 24, No. 2, pp. 122-126.

Emery, A. F., Johansson, O., and Abrous, A., 1987, "Radiation Heat Transfer Shapefactors for Combustion Systems," *Fundamentals and Applications of Radiation Heat Transfer, 24th National Heat Transfer Conference and Exhibition*, Smith, T. F. and Smith, A. M. eds., ASME, New York, NY, HTD-Vol. 72, pp. 119-126.

Foley, J. D. and Van Dam, A., 1982, *Fundamentals of Computer Graphics*, Addison Wesley, Reading, MA.

Fujimoto, A., Tanaka, T., and Iwata, K., 1984, "ARTS: Accelerated Ray-Tracing System," *IEEE Computer Graphics and Applications*, Vol. 4, No. 10, pp. 15-22.

Glassner, A. S., 1986, "Space Subdivision for Fast Ray Tracing," *IEEE Computer Graphics and Applications*, Vol. 5, No. 4, pp. 16-26.

Howell, J. R., 1989, "Thermal Radiation in Participating Media: The Past, the Present, and Some Possible Futures," *50th Anniversary Issue of The Journal of Heat Transfer*, pp. 1220-1229.

Maxwell, G. M., Bailer, M. J., and Goldschmidt, V. W., 1986 "Calculations of the Radiation Configuration Factor Using Ray Casting," *Computer-aided Design*, Vol. 18, No. 7, pp. 371-379.

Newman, W. M., and Sproull, R. F., 1979, *Principles of Interactive Computer Graphics*, McGraw-Hill, New York, New York.

O'Neill, R. F., and Zich, J. L., 1985, "Vector Sweep Engineering and Programmers Guide," Report N. GDSS-SP-85-010, General Dynamics Space Systems Division.

Walton, George N., 1987, "Algorithms for Calculating Radiation View Factors Between Plane Convex Polygons with Obstructions," *Fundamentals and Applications of Radiation Heat Transfer, Proceedings of the 24th National Heat Transfer Conference and Exhibition*, Smith, T. F. and Smith, A. M. eds., ASME, New York, NY, HTD-Vol. 72, pp. 45-52.

Annual Progress Report for ECS-8909251
Attachment #2
Summary page from "An Analysis and Modification of Shao's Radiosity Method for Computer Graphics Image Synthesis", Masters Thesis by David E. Hall

SUMMARY

This thesis begins with a history and discussion of the role of illumination models in computer graphics realistic image synthesis. The radiosity method, which is based on the principles of radiant heat transfer, is the illumination model which is the focus of the thesis. The original matrix method and the progressive refinement method for diffuse radiosity are discussed along with several implementations of the hemi-cube algorithm for form factor calculation. Accelerated implementations of the hemi-cube algorithm which utilize computer graphics hardware, spatial coherence, and pixel coherence are compared to the original software implementation, and speedups ranging from 2.1 to 5.3 are reported. Extensions of the radiosity method to include non-diffuse surfaces as performed by Immel, Rushmeier, Shao, and Sillion are presented and discussed in terms of accuracy and speed. A "correct" derivation of Shao's method is presented along with a radiosity method which successfully incorporates Shao's procedural iteration approach into a progressive refinement scheme to produce an algorithm that interactively accounts for the effects of specular surfaces on global illumination. A method to adaptively discretize specular surfaces during this progressive refinement algorithm is also presented. Several images generated using this method are shown, and timings which compare the speed of the algorithm with the original progressive refinement algorithm are given. Finally, a bidirectional approach to solving the radiosity problem in progressive refinement is presented which utilizes the concept of a "reflectance hemisphere" to determine and store the outgoing directional characteristics of the energy leaving a bidirectional surface. Results and images for this implementation are given.

E25-483

OMB Number 345-0058

NATIONAL SCIENCE FOUNDATION
1800 G STREET, NW
WASHINGTON, DC 20550

BULK RATE
POSTAGE & FEES PAID
National Science Foundation
Permit No. G-69

PI/PD Name and Address

Holly E. Rushmeier
Rm. B-146, Bldg. 225
NIST
Gaithersburg, MD 20899

NATIONAL SCIENCE FOUNDATION
FINAL PROJECT REPORT

PART I - PROJECT IDENTIFICATION INFORMATION

1. **Program Official/Org.** George Lea/ Computational Engineering

2. **Program Name** Research Initiation Awards

3. **Award Dates (MM/YY)** **From:** 9-15-89 **To:** 2-29-92

4. **Institution and Address**

The George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0405

5. **Award Number** ECS-8909251

6. **Project Title** "Progressive Refinement Algorithms for Radiant Transfer"

This Packet Contains
NSF Form 98A
And 1 Return Envelope

NSF Grant Conditions (Article 17, GC-1, and Article 9, FDP-II) require submission of a Final Project Report (NSF Form 98A) to the NSF program officer no later than 90 days after the expiration of the award. Final Project Reports for expired awards must be received before new awards can be made (NSF Grants Policy Manual Section 677).

Below, or on a separate page, provide a summary of the completed projects and technical information and attach it to this form. Be sure to include your name and award number on each separate page. See below for more instructions.

PART II - SUMMARY OF COMPLETED PROJECT (for public use)

The summary (about 200 words) must be self-contained and intelligible to a scientifically literate reader. Without restating the project title, it should begin with a topic sentence starting the project's major thesis. The summary should include, if pertinent to the project being described, the following items:

- **The primary objectives and scope of the project**
- **The techniques or approaches used only to the degree necessary for comprehension**
- **The findings and implications stated as concisely and informatively as possible**

see attached sheet

PART III - TECHNICAL INFORMATION (for program management use)

List references to publications resulting from this award and briefly describe primary data, samples, physical collections, inventions, software, etc. created or gathered in the course of the research and, if appropriate, how they are being made available to the research community.

see attached sheet

	1-23-92
Principal Investigator/Project Director Signature	**Date**

IMPORTANT:
MAILING INSTRUCTIONS
Return this *entire* packet plus all attachments in the envelope attached to the back of this form. Please copy the information from Part I, Block I to the *Attention line* on the envelope.

Holly E. Rushmeier
ECS-8909251

Part II - Summary of Completed Project

The goal of this project was to develop a progressive method for computing radiant transfer solutions for computer graphics image synthesis and radiant heat transfer applications. Progressive methods begin with a crude estimate of the solution which can be computed very quickly, and then continue indefinitely improving the accuracy.

The method for radiant transfer begins with a progressive refinement radiosity solution using a hemi-cube algorithm for form factor calculations. An improved hemi-cube algorithm was developed by exploiting computer graphics hardware capabilities and geometric coherence. A method for speeding up the radiosity solution for non-Lambertian surfaces was developed by using adaptive subdivision of non-Lambertian surfaces and the concept of a reflectance hemisphere. A Monte Carlo method was developed for calculating more detailed, accurate solutions, using the radiosity solution as a preprocess. The Monte Carlo solution proceeds in several passes to smoothly transition from crude to more accurate solutions. The overall speed of the hybrid radiosity/Monte Carlo method was increased by developing a method of simplifying complex geometries for use in the initial radiosity solution.

The radiosity portions of the method are directly applicable to both computer graphics and heat transfer. The Monte Carlo solutions have only been developed for image synthesis, but potentially could be adapted for heat transfer applications.

Holly E. Rushmeier
ECS-8909251

Part III - Technical Information

The following publications resulted from this award:

1. Hall, David E. "An Analysis and Modification of Shao's Radiosity Method for Computer Graphics Image Synthesis," MS Thesis, the George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, March 1990. Abstract published in "Dissertation Abstracts in Computer Graphics," *Computer Graphics,* July 1991.

2. Rushmeier, Holly E., Daniel R. Baum and David E. Hall. "Accelerating the Hemi-cube Algorithm for Calculating Radiation Form Factors," Radiation Heat Transfer: Fundamentals and Applications (Proceedings of AIAA/ASME Thermophysical and Heat Transfer Conference, Seattle, WA, June 18-20, 1990) HTD, Vol. 137, pp. 45-52. Revised paper in the *Journal of Heat Transfer,* Vol. 113, No. 4, November 1991, pp. 1044-1047.

3. Chen, Shenchang E., Holly E. Rushmeier, Gavin Miller, and Douglass Turner. "A Progressive Multi-Pass Method for Global Illumination," *Computer Graphics* (Proceedings of SIGGRAPH 1991, Las Vegas, NV) Vol. 25, No. 4, July 1991, pp. 165-174.

4. Hall, David E. and Holly E. Rushmeier, "An Improved Explicit Radiosity Method for Calculating Non-Lambertian Reflections," Georgia Tech Graphics, Visualization and Usability Center Tech Report GIT-GVU-91-16, submitted for journal publication.

5. Rushmeier, Holly E., Charles Patterson, and Aravindan Veerasamy, "Geometric Simplification for Indirect Illumination," submitted to SIGGRAPH '92.

The results of this project are methods and algorithms, which are being distributed to the research community by means of the publications listed above. The software developed to test these methods is not in a form suitable for general distribution, and will only be used for future research by people involved in this project.

PART IV — FINAL PROJECT REPORT — SUMMARY DATA ON PROJECT PERSONNEL

(To be submitted to cognizant Program Officer upon completion of project)

The data requested below are important for the development of a statistical profile on the personnel supported by Federal grants. The information on this part is solicited in response to Public Law 99-383 and 42 USC 1885C. All information provided will be treated as confidential and will be safeguarded in accordance with the provisions of the Privacy Act of 1974. You should submit a single copy of this part with each final project report. However, submission of the requested information is not mandatory and is not a precondition of future award(s). Check the "Decline to Provide Information" box below if you do not wish to provide the information.

Please enter the numbers of individuals supported under this grant.
Do not enter information for individuals working less than 40 hours in any calendar year.

	Senior Staff		Post-Doctorals		Graduate Students		Under-Graduates		Other Participants[1]	
	Male	Fem.	Male	Fem.	Male	Fem.	Male	Fem.	Male	Fem.
A. Total, U.S. Citizens		1			2					
B. Total, Permanent Residents										
U.S. Citizens or Permanent Residents[2]:										
American Indian or Alaskan Native ...										
Asian ...										
Black, Not of Hispanic Origin ...										
Hispanic ...										
Pacific Islander ...										
White, Not of Hispanic Origin ...		1			2					
C. Total, Other Non-U.S. Citizens										
Specify Country 1.										
2.										
3.										
D. Total, All participants (A + B + C)		1			2					
Disabled[3]										

☐ Decline to Provide Information: Check box if you do not wish to provide this information (you are still required to return this page along with Parts I-III).

[1]Category includes, for example, college and precollege teachers, conference and workshop participants.

[2]Use the category that best describes the ethnic/racial status for all U.S. Citizens and Non-citizens with Permanent Residency. *(If more than one category applies, use the one category that most closely reflects the person's recognition in the community.)*

[3]A person having a physical or mental impairment that substantially limits one or more major life activities; who has a record of such impairment; or who is regarded as having such impairment. *(Disabled individuals also should be counted under the appropriate ethnic/racial group unless they are classified as "Other Non-U.S. Citizens.")*

AMERICAN INDIAN OR ALASKAN NATIVE: A person having origins in any of the original peoples of North America, and who maintain cultural identification through tribal affiliation or community recognition.

ASIAN: A person having origins in any of the original peoples of East Asia, Southeast Asia and the Indian subcontinent. This area includes, for example, China, India, Indonesia, Japan, Korea and Vietnam.

BLACK, NOT OF HISPANIC ORIGIN: A person having origins in any of the black racial groups of Africa.

HISPANIC: A person of Mexican, Puerto Rican, Cuban, Central or South American or other Spanish culture or origin, regardless of race.

PACIFIC ISLANDER: A person having origins in any of the original peoples of Hawaii; the U.S. Pacific Territories of Guam, American Samoa, or the Northern Marianas; the U.S. Trust Territory of Palau; the Islands of Micronesia or Melanesia; or the Philippines.

WHITE, NOT OF HISPANIC ORIGIN: A person having origins in any of the original peoples of Europe, North Africa, or the Middle East.

THIS PART WILL BE PHYSICALLY SEPARATED FROM THE FINAL PROJECT REPORT AND USED AS A COMPUTER SOURCE DOCUMENT. DO NOT DUPLICATE IT ON THE REVERSE OF ANY OTHER PART OF THE FINAL REPORT.

Final Report
for
Grant ECS-8909251
"Progressive Refinement Algorithms
for Radiant Transfer"

Holly E. Rushmeier

February 1, 1992

1 Introduction

The goal of this project was to develop progressive refinement methods for the calculation of radiant transfer for applications in computer graphics and radiation heat transfer. In a progressive refinement method the solution progresses from a crude, quickly calculated solution to a refined, highly accurate solution. The project was successful in developing several new algorithms which together compose a hybrid radiosity/Monte Carlo method for calculating radiant transfer. The work was primarily performed by the principle investigator and graduate students supported by the grant. Some portions of the project were performed in collaboration with a researchers at Silicon Graphics Incorporated and at the Apple Computer Company.

2 Personnel

Two graduate students were supported by this grant – David Hall and Charlie Patterson. Other people who collaborated in the work who did not receive funds from the grant included Dan Baum, Eric Chen and Aravindan Veerasamy.

David Hall is a graduate student in the School of Mechanical Engineering at Georgia Tech. He was supported by the grant from Sept. 1989 to March 1990, when he completed his Masters' Thesis. David's work concerned the radiosity portion of the solutions. His thesis was named one of four outstanding Master's theses for the year by the Georgia Tech chapter of Sigma Xi. After completing the M.S. David went to work in private industry. In the fall of 1991 he returned to school to pursue a doctoral degree. Since the departure of the P.I. from Georgia Tech at the end of fall quarter 1991, David is now doing research in Mechanics of Materials with Dr. D. McDowell.

One portion of David's work involved the rapid calculation of form factors. He developed and implemented a new algorithm by working with the P.I. and indirectly with Dan Baum of Silicon Graphics. Dan's major contribution was insight into how calculations previously performed in software could be performed in hardware.

Charlie Patterson is a graduate student in the College of Computing at Georgia Tech. He was supported by the grant from April 1990 to August 1991. He completed a non-thesis M.S. degree, and is currently pursuing a Ph.D. Most of Charlie's work has been concerned with the development of the Monte Carlo solver that uses a radiosity solution with a simplified geometry as a preprocess. Charlie is now continuing his studies in computer graphics with Dr. Brian Guenther of the College of Computing.

Initially the P.I. developed a simple form of the hybrid radiosity/Monte Carlo solver. Eric

Chen, a member of the Advanced Technology Group at Apple Computer used the techniques in the hybrid solver combined with additional ideas for efficient sampling to produce a more sophisticated solution procedure.

One of the results of this collaboration was a grant from Apple to support continued work in this area. Aravindan Veerasamy, a Ph.D. student in the College of Computing was supported for a year on this additional grant. Aravindan developed geometries and geometry filters which were used as input to the software developed by Charlie Patterson. Aravindan is now pursuing research concerning the use of hypermedia databases for geographic information systems with Dr. Navathe of the College of Computing.

Funds from ECS-8909251 were also used to for a total of three months summer salary (over a period of two summers) and approximately one month academic year salary for the P.I. Travel funds were used for the P.I. to attend the AIAA/ASME Thermophysical and Heat Transfer Conference in Seattle, WA, June, 1990 and SIGGRAPH '91 in Las Vegas, Nevada, July, 1991.

3 Facilities

The work was primarily performed using a Silicon Graphics Personal Iris, purchased with faculty start-up funds. Funds from ECS-8909251 were used to pay maintenance fees for the machine. Initially the machine was located in the College of Engineering CAE/CAD facility. In the Fall of 1990 the machine was moved to the new Graphics, Visualization and Usability Center located in the College of Computing. In the new facility additional SGI workstations, Sun workstations, an Apple Macintosh, a film recorder and a variety of printers were available to students working on the project.

4 Summary of Results

The results of this project are detailed in the publications appended to this report. Below is a summary of each publication:

1. Rushmeier, Holly E., Daniel R. Baum and David E. Hall."Accelerating the Hemi-cube Algorithm for Calculating Radiation Form Factors," Radiation Heat Transfer: Fundamentals and Applications (Proceedings of AIAA/ASME Thermophysical and Heat Transfer Conference, Seattle, WA, June 18-20, 1990) HTD, Vol. 137, pp. 45-52. Revised paper in the *Journal of Heat Transfer,* Vol. 113, No. 4, November 1991, pp. 1044-1047.

The first portion of the overall progressive refinement solution is a progressive refinement radiosity solution. A critical factor in the speed of a radiosity solution is the time required to compute form factors. In 1985 Cohen and Greenberg (*Computer Graphics* **19(3)**:31-40) published the hemi-cube algorithm. In this paper we describe the effect of using graphics hardware to do the geometric projections required by the hemi-cube algorithm. Even though the projections account for a large fraction of the execution time of a software implementation of the algorithm, the actual speedups obtained were limited to factors on the order of 3. Other critical factors in algorithm performance are the number of surfaces in the environment, and the hemi-cube pixel resolution. The dependence on the number of surfaces was reduced by exploiting object space coherence. The surfaces are sorted into a spatial grid so that a smaller subset of surfaces to be projected onto each hemi-cube face can be easily identified. The dependence on pixel resolution was reduced by exploiting screen space coherence. Additions to form factors are performed only when different surfaces are viewed through neighboring pixels, rather than at every pixel.

2. Chen, Shenchang E., Holly E. Rushmeier, Gavin Miller, and Douglass Turner. "A Progressive Multi-Pass Method for Global Illumination," *Computer Graphics* (Proceedings of SIGGRAPH 1991, Las Vegas, NV) Vol. 25, No. 4, July 1991, pp. 165-174.

The multi-pass method in this paper describes the overall progressive refinement method for generating a realistic computer image. In the first pass, a radiosity method gives the first, quickly produced solution. In the second pass,a more detailed solution for the direct illumination is found by replacing the direct illumination component in the radiosity solution by a pixel by pixel Monte Carlo calculation. Finally, in a third pass the indirect illumination is calculated in greater detail by using a Monte Carlo method which uses the radiosity solution to estimate the effect of higher order interreflections.

3. Hall, David E. and Holly E. Rushmeier, "An Improved Explicit Radiosity Method for Calculating Non-Lambertian Reflections," Georgia Tech Graphics, Visualization and Usability Center Tech Report GIT-GVU-91-16, submitted for journal publication.

Radiosity methods were originally developed for Lambertian (ideal diffuse) surfaces. In 1988, Shao et al. (*Computer Graphics* **22(3)**:93-101) presented a radiosity method that could account for the effect of non-Lambertian reflections. In this paper a progressive refinement implementation of Shao's method is described. Two improvements to Shao's method

are proposed. First, the subdivision of specular surfaces is examined. Since the radiances calculated for specular surfaces are not viewed in the final image, adaptive subdivision of specular surfaces based on their effect on Lambertian surfaces is used. Second, the large number of computations to compute reflected radiance distributions from non-Lambertian surfaces is examined. The number of computations is reduced for non-Lambertian/non-specular surfaces by exploiting the averaging effect of reflection from such surfaces.

4. Rushmeier, Holly E., Charles Patterson, and Aravindan Veerasamy, "Geometric Simplification for Indirect Illumination," submitted to SIGGRAPH '92.

Although the multi-pass method described in reference 2. above is effective, as scenes become complex (i.e. contain large numbers of surfaces) the time to compute the radiosity solution becomes excessive. In this paper a method for simplifying a complex scene for the radiosity pass of the calculations is proposed. The scene is simplified by eliminating small, isolated surfaces, and by replacing clusters of small surfaces with a box that is approximately equivalent in terms of the radiant energy reflected and absorbed. The scene complexity becomes apparent in the Monte Carlo direct and indirect illumination passes, for which computation time is a weaker function of the scene complexity.

5 Future Work

This project successfully developed an overall progressive refinement solution method for radiant transfer, and developed improvements in several elements of the solution method. There are two major points discussed in the initial proposal which were not examined.

First is the examination of the impact on accuracy of using multiple levels of geometric description. So far, only a few visual comparisons of the images produced using the multi-pass method with geometric simplification have been performed. Quantitative comparisons for a variety of environments are needed to assess the method.

The second point is the application of the techniques developed in this project to radiant heat transfer. The improved hemi-cube algorithm can be used directly in heat transfer calculations. The application of the progressive refinement method for non-Lambertian surfaces is straightforward. An analysis is need however of the accuracy of the results as a function of the hemi-cube and reflectance hemisphere used. The usefulness of hybrid methods in heat transfer is not as clear. The "inverse" Monte Carlo and hybrid methods used in image synthesis might be useful in heat transfer to get highly accurate results at discrete points to compare with the area averaged results obtained by forward Monte Carlo or radiosity methods.

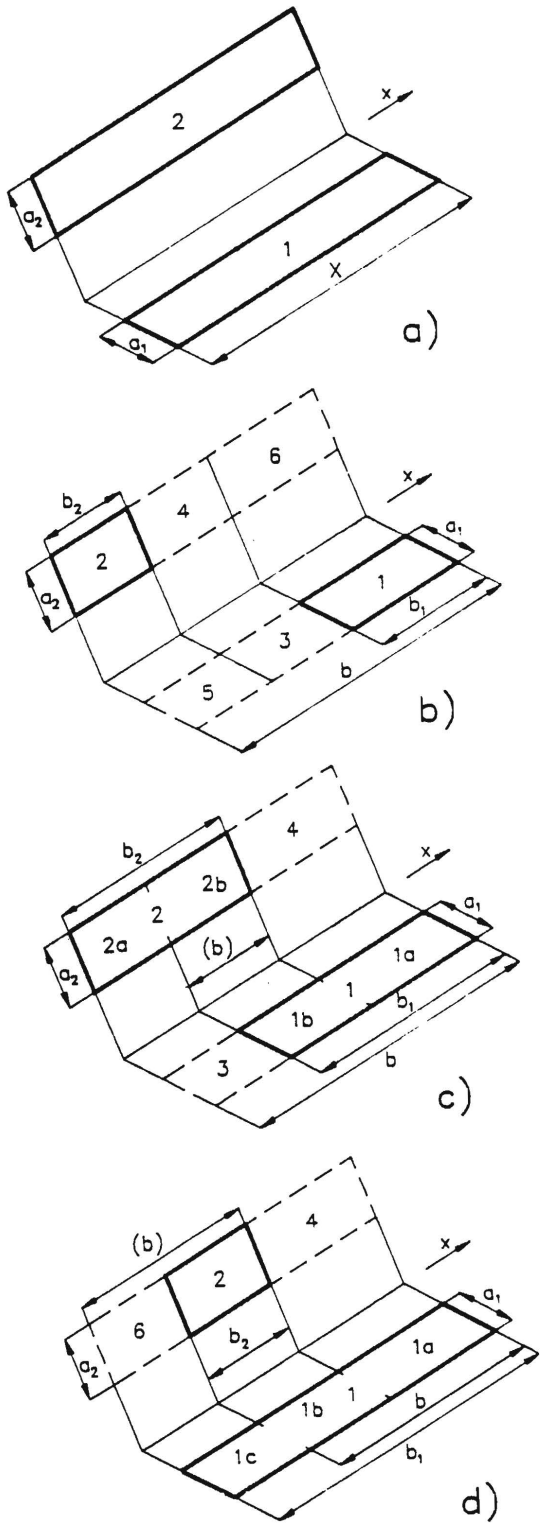


Fig. 3 Scheme for derivation of generalizing rule (A5)

$$\overline{S_1S_2} = \frac{1}{2}[\overline{SS_o}(b) - \overline{SS_o}(|b-b_1|) - \overline{SS_o}(|b-b_2|) + \overline{SS_o}(|b-b_1-b_2|)] \quad \text{(A5)}$$

With overlapping zones, we can, according to Fig. 3(*c*) and Fig. 3(*d*), alternatively substitute into (A5) for distance b the dimension marked "(*b*)". Using it, the same result will be reached.

The derived simple generalized rule (A5) is independent of the absorptivity function of the medium between the zones and is valid also, e.g., for two parallel cylinders, band-to-rectangular rod, etc. This rule can be applied not only to pairs of surface zones, but similarly to direct-exchange areas of volume-to-surface zones $\underline{gs}$ and volume-to-volume zones $\underline{gg}$.

Accelerating the Hemi-Cube Algorithm for Calculating Radiation Form Factors

H. E. Rushmeier,[1] D. R. Baum,[2] and D. E. Hall[3]

Introduction

The calculation of form factors for the analysis of radiation heat transfer is computationally expensive when surfaces are not in full view of one another. For an enclosure of N surfaces, N^2 factors are needed. Brute force methods require $O(N^3)$ time to compute the required factors (Walton, 1987). A few methods have been introduced that reduce the time to $O(N^{2+x})$ where x is on the order of 0.3 (Emery et al., 1988). The hemi-cube algorithm (Cohen and Greenberg, 1985) calculates the N^2 factors in $O(N^2)$ time. Because of its lower time complexity, the hemi-cube method is efficient for problems with large numbers of surfaces. In this note, methods to improve the hemi-cube algorithm by taking advantage of graphics hardware and two types of geometric coherence are presented. The implementation of the methods on a computer workstation is described, and timings demonstrating speedups by factors of up to 6.7 are given.

The acceleration of the calculation of form factors is not an isolated computational problem. The ability to calculate form factors very quickly can alter the overall computational methodology used to analyze radiant heat transfer, and can greatly extend the complexity of geometries that can be studied.

Review of the Hemi-Cube Algorithm

The hemi-cube algorithm approximates the form factor F_{ij} from an area A_i to area A_j by the factor from a differential area dA_i at the center of area A_i:

$$F_{ij} \cong \int_{A_j} \cos\theta_i \cos\theta_j \, dA_j / \pi r_{ij}^2$$

where the angles θ_i and θ_j are measured respectively from the surface normals at dA_i and dA_j to a line joining the two differential areas, and r_{ij} is the distance between dA_i and dA_j. Surfaces in the enclosure are subdivided to make this approximation valid.

The hemi-cube method is based on Nusselt's unit sphere method. The hemisphere in Nusselt's method is replaced by a hemi-cube, as shown in Fig. 1. The sides of the hemi-cube are

[1] George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0405; Assoc. Mem. ASME.
[2] Silicon Graphics Computer Systems, Mountain View, CA 94039-7311.
[3] George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0405.

Contributed by the Heat Transfer Division of The American Society of Mechanical Engineers. Manuscript received by the Heat Transfer Division September 12, 1990; revision received April 2, 1991. Keywords: Numerical Methods, Radiation.

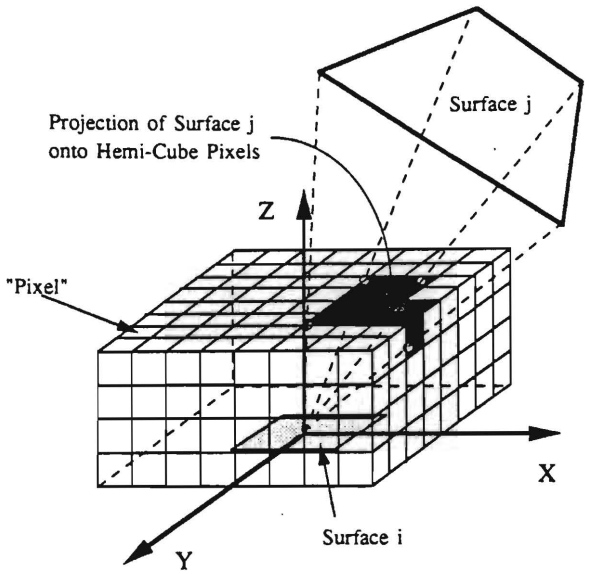


Fig. 1 Calculation of F_{ij} using a low-resolution hemi-cube

discretized into small surfaces, referred to as pixels. The form factor to each pixel, referred to as a delta-form factor, is calculated and stored in a buffer. Since the same hemi-cube is used for all surfaces, the values in the delta-form factor buffer are calculated only once.

The surface visible through each hemi-cube pixel for each differential area dA_i is found efficiently using backface elimination, the Sutherland–Hodgeman clipping algorithm, and the depth buffer algorithm. (The details of these techniques can be found in standard computer graphics texts such as Foley et al., 1990.) The result is an item buffer, in which the identifier of the surface visible through each pixel is recorded.

After the hidden surface problem has been solved, the form factors are calculated by scanning the item buffer and summing the delta form factors associated with the pixels through which a surface j is visible.

The hemi-cube algorithm is summarized in the following pseudocode:

```
FOR each hemi-cube pixel q calculate DeltaFactor(q);
FOR each surface i
  FOR each hemi-cube face s
    FOR each surface j
      FormFactor(ij) = 0;
      Transform coordinates to system centered on i and
      oriented toward s;
      IF (not back facing)
        Clip to viewing frustum;
        IF (portion of j inside the frustum)
          Find pixels p onto which j is projected;
          FOR each pixel p
            IF (depth of j < depth (p))
              item (p) = j;
              depth (p) = depth of j;
            END IF
          END FOR each pixel
        END IF portion left
      END IF not back facing
    END FOR each surface j
  END FOR each hemi-cube face
  FOR each pixel q on the hemi-cube FormFactor(i,
  item(q)) + = DeltaFactor(q);
END FOR each surface i.
```

The $O(N^2)$ time complexity of the algorithm is achieved by determining the effect of each surface j as an obstructor of other surfaces as the factor F_{ij} is being computed. Inherently, any algorithm that simultaneously computes visibility and form factors cannot take advantage of the reciprocity relation used in $O(N^3)$ and $O(N^{2+x})$ algorithms. However, for large values of N the factor of 2 speedup from reciprocity is much smaller than the factor of N or N^x time difference between the hemi-cube algorithm and alternative methods.

The accuracy of the hemi-cube algorithm is discussed in more detail elsewhere (Baum et al., 1989). Overall, the method converges to an exact solution with the discretization of the enclosure into smaller surfaces, and the discretization of the hemi-cube into a larger number of pixels.

Acceleration of the Hemi-Cube Method Using Hardware

Recently computer graphics workstations have been introduced by several vendors with specialized graphics processors, which perform many of the operations required by the hemi-cube algorithm in hardware rather than software. A library of subroutine calls to the graphics hardware is provided with the workstation.

The use of the graphics hardware in the hemi-cube algorithm begins by allocating a window on the graphics screen to represent a face on the hemi-cube. The surfaces are then displayed in this window by sending the coordinates of the vertices of each surface to the graphics processor. The integer identifier of each surface is sent to the graphics processor as the color to be displayed. After all the surfaces have been displayed, the item buffer is filled by reading the color being displayed in the graphics window at each pixel.

Using graphics hardware, the pseudo-code for the loop over each surface i becomes:

```
Allocate a window on the workstation to be used as a face
of the hemi-cube;
FOR each surface i
  FOR each hemi-cube face s
    FOR each surface j
      FormFactor(ij) = 0;
      Color = j;
      Send a list of vertices for surface j to the graphics
      processor;
    END for each surface j
  END FOR each hemi-cube face
  Read the workstation window color buffer into the item
  buffer;
  FOR each pixel q on the hemi-cube FormFactor(i,
  item(q)) + = DeltaFactor(q);
END FOR each surface i.
```

Typical timing results for the software and hardware implementations of the hemi-cube algorithm are shown in Table 1. All timings are for a Silicon Graphics Personal Iris W-4D20G workstation. The timings include only the time to compute the N^2 form factors, not the time to write the factors to a file, or to perform radiant transfer calculations.

Results are shown for two geometries. Geometry 1 consists of a cubical enclosure 1 unit on each side, which contains a uniformly distributed three-dimensional array of 64 cubes, each 0.1 unit on a side. Geometry 2 is the same enclosure containing two cylinders. One cylinder is centered at (0.7, 0.5, 0.5), has radius 0.1, height 0.3, and is tilted 20 deg. The other cylinder is centered at (0.3, 0.4, 0.6), has radius 0.2, height 0.4, and is tilted 30 deg. Each cylinder is discretized into 140 planar polygons. Results are given for two different levels of hemi-cube discretization. In Table 1 the speedup obtained using the hardware ranges from 1.6 for Geometry 1 using a low resolution hemi-cube to 3.5 for Geometry 2 using a higher resolution hemi-cube.

Once the vertices of a surface have been sent to the graphics processor, the operations of back face elimination, clipping and updating the depth buffer are spedup by a factor of 100

or more. However, since parts of the algorithm are still performed in software, the overall speed up obtained by using hardware is much lower. In the hardware implementation of the hemi-cube algorithm, the operations of sending the vertices to the graphics processor and adding up the delta form-factors account for essentially all of the time to compute factors.

Accelerating the Hemi-Cube Using Spatial Coherence

The time spent sending vertices to the graphics processor can be reduced by taking advantage of spatial coherence. Groups of polygons that lie behind the current polygon can be identified, so that individual polygons in these groups do not have to be processed. Polygons can be divided into groups by sorting them into a fixed spatial grid. Polygon vertices are sent to the graphics processor only if they lie in grid elements that lie in front of the current polygon. The sorting of polygons into a spatial grid is similar to the spatial subdivision techniques used to accelerate ray tracing in computer graphics (Glassner, 1984; Fujimoto et al., 1986). Also, Emery et al. (1987), using an idea presented by Hedgeley (1982), developed a method in which surfaces were sorted into a spatial grid to identify potential obstructors between a pair of surfaces rapidly.

The pseudocode for this variation of the algorithm is:

```
FOR each surface i
  Determine which volumes in the spatial grid contain
  surface i;
  FOR each volume v which contains i add i to list of
  surfaces associated with v;
END FOR surface i
Allocate a workstation window to be used as the hemi-cube
face;
FOR each surface i
  FOR each surface j Front(j) = 0
  FOR each volume v
    IF (v or part of v is in front of i)
      FOR each surface k on the list associated with v
      Front(k) = 1;
  END FOR each volume;
  FOR each hemi-cube face s
    FOR each surface j
      FormFactor(i, j) = 0;
      Color = j;
      IF (Front(j) = = 1) send a list of vertices for surface
      j to the graphics processor;
    END FOR each surface j
  END FOR each hemi-cube face
  (continue as before)
END FOR each surface i.
```

Typical timing results for the spatial coherence variation of the hemi-cube algorithm are given in Table 1. The overall speedup over the original software implementation of the hardware version with spatial coherence ranges from a factor of 1.9 to a factor of 3.6 for the results in Table 1. The incremental speedup over of the hardware version without coherence ranges from little more than 1.0 to 1.2. Although the factors are small, Table 1 shows that using spatial coherence always reduced the total computational time.

The spatial coherence method could also be applied to a purely software implementation of the hemi-cube. In the software implementation both the absolute time and the fraction of time spent projecting each surface onto the hemi-cube are different from the hardware version. As a result, both the absolute time saved and the factor of speedup for using spatial coherence would be different when applied to the software implementation.

Greater speedups could be obtained by determining which grid volumes will be visible through each of the five hemi-cube faces and further limiting the number of vertices that have to be sent to the graphics processor. The average number of vertices sent to the graphics processor for each screen could be reduced by as much as a factor of 5. The overall speedup of the form factor calculation produced by this improvement would still depend on the fraction of the overall computation time spent sending vertices to the processor.

Table 1 Timings for form factor calculations on a SGI Personal Iris workstation (in CPU seconds)

Geometry	Hemi-cube res., R*	Method**	Time	Overall speedup***	Coherence speedup****
1	50	SW	118	—	—
		HW	72	1.6	—
		HWSC	60	2.0	1.2
		HWSCPC	54	2.2	1.3
1	300	SW	1353	—	—
		HW	387	3.5	—
		HWSC	380	3.6	1.0
		HWSCPC	201	6.7	1.9
2	50	SW	66	—	—
		HW	42	1.6	—
		HWSC	35	1.9	1.2
		HWSCPC	32	2.1	1.3
2	300	SW	855	—	—
		HW	282	3.0	—
		HWSC	276	3.1	1.0
		HWSCPC	141	6.1	2.0

* Number of hemi-cube pixels = $R^2 + 4(0.5)(0.6R \times 0.6R)$
** SW = Software implementation
HW = Hardware implementation
HWSC = Hardware implementation with spatial coherence
HWSCPC = Hardware implementation with spatial coherence and pixel coherence
*** Overall speedup = (Time for software implementation)/(Time)
**** Coherence speedup = (Time for HW)/(Time)

Accelerating the Hemi-Cube Using Pixel Coherence

Surfaces projected onto a hemi-cube tend to cover contiguous sets of pixels. Adding the delta-form factors can be sped up using this coherence by updating the value of the form factor for continuous runs of hemi-cube pixels, rather than for every pixel. Instead of storing a delta-form factor for each pixel, the cumulative form factor is stored, i.e., CumulativeFactor (q) is equal to the sum of all DeltaFactor (q') factors for pixels 1 to q. In the loop to add delta-form factors, an addition is performed only when the value in the item buffer is different for two neighboring pixels, i.e;

```
Start = item(1);
OldValue = 0.
FOR each pixel q on the hemi-cube
  IF (item(q) not equal to Start):
    TempValue = CumulativeFactor(q - 1);
    FormFactor(i, Start) + = TempValue-OldValue;
    Start = item(q);
    OldValue = TempValue;
  END IF item (q)
END FOR each pixel.
```

Results for the pixel coherence variation are shown in Table 1. The overall speedup over the original software implementation of the hardware version with spatial and pixel coherence ranges from 2.1 to 6.7. The incremental speedup over the basic hardware implementation ranges from 1.3 to 2.0. The speedup depends on the average number of pixels through which each polygon is viewed. Accuracy requires that each surface be visible through more than one pixel, and so this variation will always result in some speedup.

The pixel coherence variation can also be applied to the purely software implementation. In the software implementation the same absolute time is spent summing delta-form factors as in the hardware version, but this time is a smaller percentage of the overall computation time. As a result, the absolute time saved using pixel coherence would be the same

for both the software and hardware implementations but the factor of speedup would be different.

Summary

Three methods to accelerate the hemi-cube method for finding form factors have been presented: use of graphics hardware, use of spatial coherence and use of pixel coherence. The overall speedup resulting from these methods depends on the particular enclosure studied, and the required accuracy. Results have been presented illustrating speedups of factors up to 6.7.

Acknowledgments

This work was funded, in part, by NSF grant ECS-8909251, "Progressive Refinement Algorithms for Radiant Transfer."

References

Baum, D. R., Rushmeier, H. E., and Winget, J. M., 1989, "Improving Radiosity Solutions Through the Use of Analytically Determined Form-Factors," *ACM Computer Graphics (Proceedings of SIGGRAPH 89)*, Vol. 23, No. 3, pp. 325–334.

Cohen, M. F., and Greenberg, D. P., 1985, "The Hemi-Cube: A Radiosity Solution for Complex Environments," *ACM Computer Graphics (Proceedings of SIGGRAPH 1985)*, Vol. 19, No. 3, pp. 31–40.

Emery, A. F., Johansson, O., and Abrous, A., 1987, "Radiation Heat Transfer Shapefactors for Combustion Systems," *Fundamentals and Applications of Radiation Heat Transfer, 24th National Heat Transfer Conference and Exhibition*, T. F. Smith and A. M. Smith, eds., ASME HTD-Vol. 72, pp. 119–126.

Emery, A. F., Johansson, O., Lobo, M., and Abrous, A., 1988, "A Comparative Study of Methods for Computing the Diffuse Radiation Viewfactors for Complex Structures," presented at the AIAA/ASME/ASCE/ASH 29th SDM Conference, Apr. 18–20, Williamsburg, VA, Paper No. AIAA-88-2223.

Foley, J. D., Van Dam, A., Feiner, S. K., and Hughes, J. F., 1990, *Computer Graphics Principles and Practices*, Addison Wesley, Reading, MA.

Fujimoto, A., Tanaka, T., and Iwata, K., 1986, "ARTS: Accelerated Ray-Tracing System," *IEEE Computer Graphics and Applications*, Vol. 6, No. 4, pp. 16–26.

Glassner, A. S., 1984, "Space Subdivision for Fast Ray Tracing," *IEEE Computer Graphics and Applications*, Vol. 4, No. 10, pp. 15–22.

Hedgeley, D. R., Jr., 1982, "A General Solution to the Hidden Line Problem," NASA Reference Pub. 1085.

Walton, G. N., 1987, "Algorithms for Calculating Radiation View Factors Between Plane Convex Polygons With Obstructions," *Fundamentals and Applications of Radiation Heat Transfer, Proceedings of the 24th National Heat Transfer Conference and Exhibition*, T. F. Smith and A. M. Smith, eds., ASME HTD-Vol. 72, pp. 45–52.

An Improved Explicit Radiosity Method for Calculating Non-Lambertian Reflections

David E. Hall
Holly E. Rushmeier
The George Woodruff School of Mechanical Engineering
and the Graphics, Visualization and Usability Center
Georgia Institute of Technology
Atlanta, GA 30332-0405

Submitted to *The Visual Computer*
August, 1991

An Improved Explicit Radiosity Method for Calculating Non-Lambertian Reflections

Abstract

We present an improved radiosity method for accounting for non-Lambertian reflections. The method explicitly calculates the radiance distribution leaving each non-Lambertian surface. The method differs from previous explicit radiosity methods in two respects. First, non-Lambertian surfaces are discretized adaptively based on their effect on other surfaces, rather than on their own spatial radiance distribution. Second, the calculation of the radiance distribution is made more efficient using the ideas of hemi-cube pixel groups and the reflectance hemi-sphere. The method is well suited to being used as the first pass in a multi-pass rendering method.

key words: radiosity, global illumination, general reflectance functions

1. Introduction

The radiosity method (Goral et al., 1984, Nishita and Nakamae, 1985) has become popular for solving the global illumination problem for computer graphics realistic image synthesis. A major shortcoming of the original method was its limitation to Lambertian (i.e. ideal diffuse) materials. Two types of methods have been developed to account for the effects of non-Lambertian (i.e. directional) materials, which we will refer to as explicit and implicit methods. Several variations of each type of method have been formulated. The best method to use depends on the particular application. The purpose of this paper is to contribute to the study of radiosity methods for general surfaces by examining two problems encountered in using explicit methods, and proposing solutions to these problems. The method presented here is most useful as a fast initial radiosity pass for a multi-pass global illumination solution.

2. Background

In implicit radiosity methods, the complete radiance distributions of directional surfaces are never calculated. The effect of directionally reflecting surfaces on Lambertian surfaces is accounted for implicitly by introducing "mirror-images" of the Lambertian reflectors, or by tracing trees of rays between Lambertian surfaces. Implicit methods require a two pass approach. In the first pass the radiosities of Lambertian surfaces are calculated. Then, in a second view dependent pass, the radiances of directional surfaces in the view direction are calculated. Implicit methods have been described by several researchers, including Wallace et al. (1987), Malley (1988), Sillion and Puech (1989) and Rushmeier and Torrance (1990). More recently Chen et al. (1991) presented a multi-pass method in which an implicit radiosity method is used to get a quick first estimate of global

illumination. The radiances of both non-Lambertian and Lambertian surfaces discretized at screen resolution are calculated in subsequent passes, using information from the radiosity solution.

By contrast, in explicit methods a radiance distribution is explicitly calculated for each directional surface at some point in the solution. The first explicit method was by Immel et al. (1986). In Immel's method a large set of simultaneous equations was solved for the radiance in each direction from each surface in the environment. The spatial discretization required to model radiance distributions for highly directional surfaces made this method impractical. Shao et al. (1988) built on Immel's idea, and used the two pass idea from implicit methods. In Shao's method the radiance distribution from each directional surface is explicitly calculated to find revised form factors from non-Lambertian surfaces, which in turn are used to calculated the radiosities (and equivalently the radiances) of Lambertian surfaces. In the second pass, radiances calculated from the first pass are used to render Lambertian surfaces, but the radiances for directional surfaces are found by ray tracing. In Shao's method the level of discretization for directional surfaces in the first pass is greatly reduced since the first pass directional radiance distributions are not seen directly in the final image. Le Saec and Schlick (1989) and Chen and Wu (1990) have presented similar explicit methods which use the progressive refinement solution method originated by Cohen et al. (1988).

All of the explicit methods described above calculate the gathering or distribution of light by discretizing the hemisphere of directions above a surface. Immel and Le Saec represent the resulting radiance distributions using the discretized directions. Shao and Chen represent the distribution in terms of form factors to other surfaces. Recently Sillion et al. (1991) presented an explicit method in which light is distributed by considering each surface vertex in the environment, rather than looking out in some discrete set of directions. The resulting radiance distributions are saved as coefficients of spherical harmonics, rather than as values associated with directions or as form factors. This framework is capable of rendering spatially detailed solutions. However, considering interreflections vertex by vertex, rather than in terms of directions is extremely costly as the geometric complexity of the environment increases (e.g. see Kajiya, 1986). In Sillion's method only mirror-like specular surfaces are calculated in the second pass. While this produces a largely "view independent" solution, the meshing requirements to avoid visual artifacts and the storage space required per vertex for such an approach are extremely high.

We consider an explicit method which is based on the discretized hemisphere, rather than the vertex by vertex approach. While this method is not as useful as Sillion's for a spatially detailed radiosity solution, it is much more efficient for use as a first pass in a multi-pass solution. We focus on two basic problems inherent with explicit, discretized direction methods-- determining the meshing of directional surfaces, and the excessive calculations required for general directional surfaces which are neither Lambertian nor mirror-like reflectors. To mesh directional surfaces, rather than using the same algorithms as for Lambertian surfaces, we propose adaptively meshing directional surfaces based on their effect on Lambertian surfaces relative to the overall scene illumination. To reduce the number of calculations required for general directional surfaces, we introduce the reflectance hemisphere and hemi-cube pixel groups to take advantage of the averaging effect of non-mirrorlike surfaces.

We begin by describing a specific progressive refinement formulation of an explicit radiosity method for including directional reflections.

3. Formulation

Progressive refinement radiosity methods work by determining the effect of the energy leaving one surface on all other surfaces in the environment.This is accomplished by "shooting" energy out from a surface, beginning with light sources shooting out their emitted energy. For each surface two values are maintained. The total energy leaving the surface and the "unshot" energy that has yet to be distributed to the rest of the environment.

To formulate a progressive refinement radiosity method for non-Lambertian surfaces, specific definitions are needed of radiance, radiosity, and reflectance. The radiance I(j) in a direction j, either incident on or leaving a surface, is defined as:

$$I(j) = dE(j)/\cos\theta d\omega \tag{1}$$

where dE(j) is the energy per unit area and time, θ is the angle between the surface normal and the direction j, and $d\omega$ is a differential solid angle. The radiosity of a surface B is the total energy per unit area and time leaving the surface. Radiosity and radiance are related by:

$$B = \int I(j) \cos\theta \, d\omega \tag{2}$$

where the integral is over the hemisphere of directions leaving j. For Lambertian surfaces, I is independent of j and B is equal to $I\pi$.

The bidirectional reflectance $\rho_{bd}(r,i)$ of a surface is the radiance reflected in direction r divided by the incident energy per unit area and time from the direction i:

$$\rho_{bd}(r,i) = I(r)/E(i) \tag{3}$$

Associated with the bidirectional reflectance is the directional-hemispherical reflectance $\rho_{dh}(i)$, defined as the energy per unit area and time reflected into the entire hemisphere of directions, divided by E(i). The relationship of $\rho_{dh}(i)$ to $\rho_{bd}(r,i)$ is obtained by integrating Eq. (3) over the reflected hemisphere.

$$\rho_{dh}(i) = \int \rho_{bd}(r,i) \cos\theta_r \, d\omega_r \tag{4}$$

For Lambertian surfaces $\rho_{bd}(r,i)$ is independent of directions r and i, and $\rho_{dh}(i)$ is denoted as ρ_d and is just equal to $\pi\rho_{bd}$. For specular surfaces, ρ_{bd} is a delta function, $\rho_s(i)\delta(r,i)/\cos\theta_r d\omega_r$ with the energy incident from direction i reflected only into the mirror direction. $\rho_{dh}(i)$ for a specular surface is just the specular reflectance $\rho_s(i)$.

Consider a general surface M shooting out energy. The hemisphere of directions above M is discretized by a hemi-cube (Fig. 1). (Note: although we use a hemi-cube as defined by Cohen and Greenberg (1985), the method we describe could be applied to any discretization of the hemisphere above a surface and any method of determining the surface visible in a particular direction.)

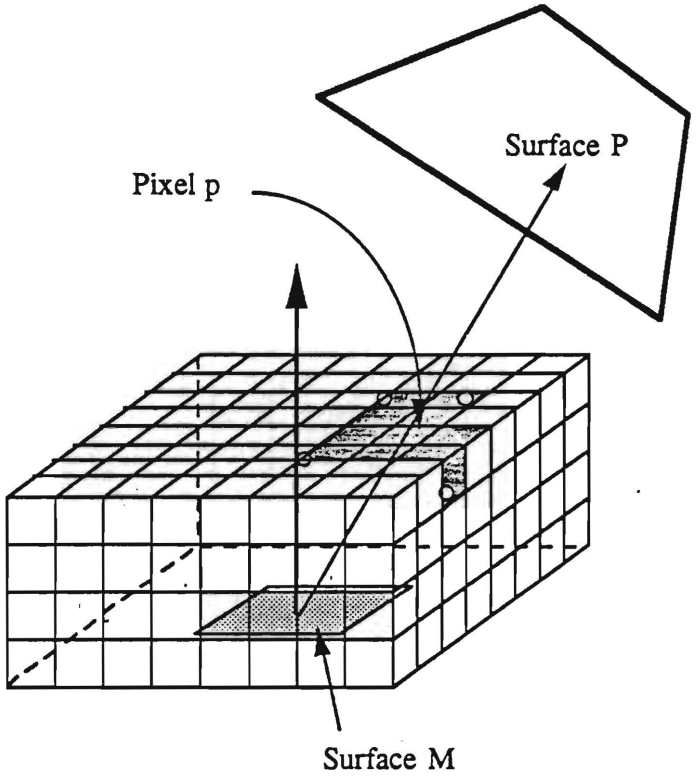


Figure 1 - The hemisphere of directions above M is discretized by a hemi-cube. Pixel p is one of the hemi-cube pixels through which surface P is visible to surface M.

The unshot radiosity of surface M is to be distributed into all directions. Associated with the unshot radiosity B_u is an unshot radiance distribution. Let the unshot radiance from M in the direction of a hemi-cube pixel p be $I_{u,M,p}$. The energy per unit area of surface M leaving through p, $E_{M,p}$ is:

$$E_{M,p} = I_{u,M,p} \cos\theta_p \Delta\omega_p = I_{u,M,p} \Delta F_p \pi \tag{5}$$

where ΔF_p is the delta form factor associated with pixel p.

Let P be the surface visible to M through pixel p, and let m be the direction from P to surface M. The delta radiosity at surface P, ΔB_P, due to the energy leaving M traveling through pixel p is:

$$\Delta B_P = \rho_{dh,P}(m)\, I_{u,M,p} \Delta F_p \pi A_M / A_P \tag{6}$$

where m is the direction from P to M, and the factor (A_M/A_P) converts from energy per unit area of surface M to energy per unit area of surface P. (Note: as described in Baum et al., 1989, $\Delta F_p A_M/A_P$ may be poorly estimated using the hemi-cube. Once the surface P is identified, $\Delta F_p A_M/A_P$ may be calculated by an alternative analytical method.)

When the shooting surface M is Lambertian, the simple relationship between radiosity and radiance gives:

$$\Delta B_P = \rho_{dh,P}(m)\, B_{u,M} \Delta F_p (A_M/A_P) \tag{7}$$

When the shooting surface M is non-Lambertian, the explicit unshot radiance distribution $I_{u,M,p}$ must be calculated taking into account the incident radiance distribution. Let Q be the surface visible through pixel q, and $I_{u,Q,M}$ be the radiance shot from Q to M which has yet to be distributed by M:

$$I_{u,M,p} = \sum \rho_{bd,M}(q,p) I_{u,Q,M}\, \Delta F_q \pi \tag{8}$$

where the summation is over all hemi-cube pixels q. For non-Lambertian surfaces the hemi-cube serves two purposes. First, it is used to calculate the energy incident on M which has yet to be distributed. Then, it is used to distribute the energy after the unshot radiance distribution from M has explicitly been calculated. For a non-Lambertian surface M, Eq. (6) becomes:

$$\Delta B_P = \rho_{dh,P(m)} \left(\sum \rho_{bd,M}(q,p) I_{u,Q,M}\, \Delta F_q \pi\right) \Delta F_p \pi A_M / A_P \tag{9}$$

Once a non-Lambertian surface shoots, all of the unshot radiances to that surface are set to zero.

Note that for the special case of specular surfaces, Eq. (9) simplifies to:

$$\Delta B_P = \rho_{dh,P(m)} (\rho_{s,M}(p) I_{u,P',M}\,) \Delta F_p \pi A_M / A_P \tag{10}$$

where P' is the surface seen through the pixel p' in the mirror direction from pixel p.

Initially, the unshot radiance from one surface to another, i.e. $I_{u,Q,M}$ can be calculated directly only for light sources. The explicit radiance distribution of the light sources must be given as part of the problem definition. The progressive refinement solution begins by shooting out energy from light

sources. As the solution proceeds the radiance $I_{u,Q,M}$ a non-Lambertian surface M receives from any surface Q must must be saved until that energy is shot out from M to the rest of the environment. Since M may be visible through several pixels from Q, the quantity to be stored when Q is also non-Lambertian is the following average:

$$I_{u,Q,M} = \Sigma\, I_{u,Q,k}\Delta F_k /(\Sigma\, \Delta F_j) \qquad (11)$$

where the summations are over all pixels (k in the numerator, j in the denominator) through which M is visible to Q. If Q is Lambertian, $I_{u,Q,M}$ is just equal to the value of $B_{u,Q}/\pi$ at the time energy was shot from Q to M.

Clearly, the disadvantage of the formulation just given is keeping all of the values $I_{u,Q,M}$ for non-Lambertian surfaces. All radiosity methods have growing storage/processing demands as the number of directional surfaces increases (i.e. increasing "bushy" ray trees are followed, or more detailed storage structures for radiance distributions are needed, etc.). Radiosity methods are efficient when nearly all surfaces can be treated as Lambertian for the purpose of computing the secondary illumination of other surfaces. For only a few non-Lambertian surfaces in the radiosity pass, the non-Lambertian surfaces can be treated efficiently as a group. When the unshot energy for the entire group of non-Lambertian surfaces is the largest of all unshot energies, all of the non-Lambertian surfaces shoot as a group. Each surface then maintains three unshot radiosities -- B_u unshot to the environment, $B_{s,nr}$ shot out to the environment but not redistributed by non-Lambertian surfaces, and $B_{u,nL}$ unshot radiosity accumulated during the shooting phase from all non-Lambertian surfaces. Values of $I_{u,Q,M}$ only need to be saved then if both Q and M are non-Lambertian.

4. Surface Discretization

The sampling of directions from a point on a surface requires discretizing surfaces. Shao's approach reduces the requirements for the discretization of directional surfaces by not using the explicit directional distribution in the final image. However, inadequate discretization of the directional surface can result in significant errors, as demonstrated in Figs. 2a and 2b. In Fig. 2a the specular surface is discretized into only 3 patches, resulting in a "gap" in the bright area on the floor resulting from the specular reflection of the light source. In Fig. 2b, the specular surface is discretized into 75 patches, and the bright area on the floor is continuous, as it should be.

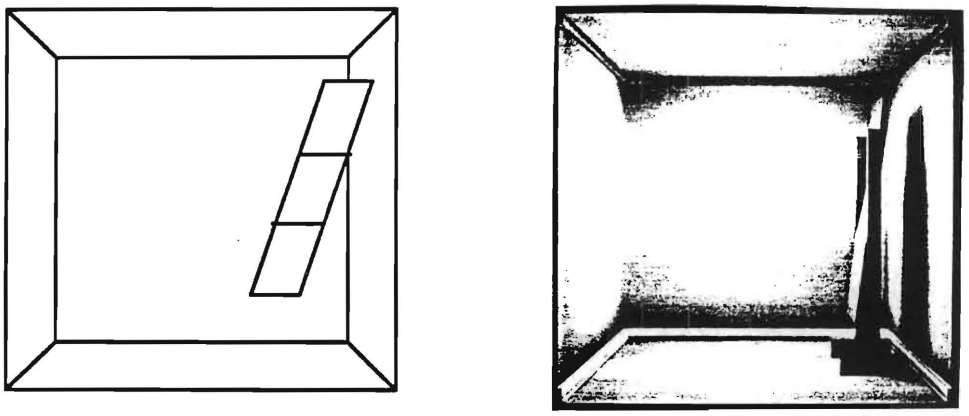


Figure 2a - Specular surface divided into 3 patches

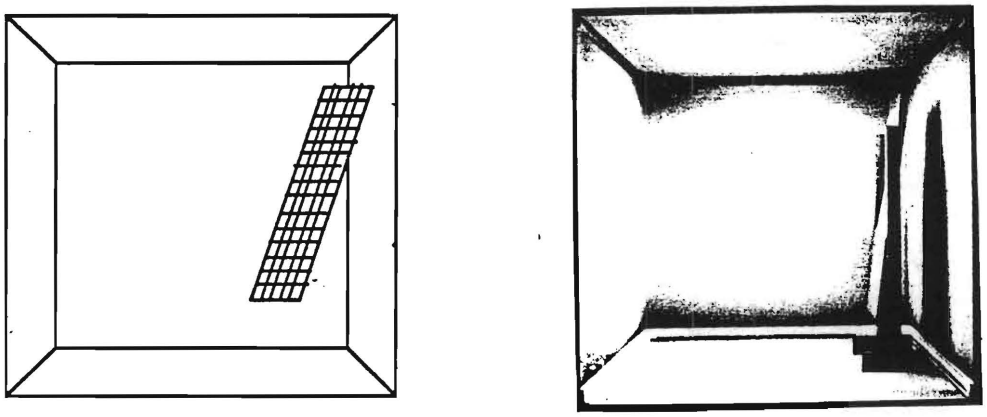


Figure 2b - Specular surface divided into 75 patches

Figure 2 - A pair of images demonstrating the effect of specular surface discretization on global illumination.

In the past, discretization of surfaces in radiosity methods has been determined by the view of the surface from the light source (Campbell & Fussell, 1990, Baum et al. 1991) followed by adaptive subdivision based on the relative radiance of neighboring surfaces (Cohen et al. 1986). Both of these methods are oriented towards subdividing a surface to make its appearance acceptable in the

final image. Since the radiosity solution for directional surfaces is not directly visible in the final image, these approaches are inappropriate. In their place, we propose an adaptive sub-division method based on the effect of the directional surface on the Lambertian surfaces in the environment. A crude initial discretization is applied to each directional surface, and the surface is subdivided until the error in the incremental radiosities for the surfaces it is shooting at converges.

The true error that results from insufficient surface discretization can not be evaluated because the true solution is not known. Instead, the error calculation is based on the relative changes in ΔB as the non-diffuse surface discretization changes. For the adaptive discretization method we propose, the error in the radiosity of any surface j is determined using the equation:

$$\text{Error} = \Delta B_{g,j} - \Delta B_{g,j+1} / \rho_j G \qquad (12)$$

where $\Delta B_{g,j}$ is the increase in unshot radiosity of a surface j at discretization level g, $\Delta B_{g+1,j}$ is the increase in unshot radiosity of surface j at discretization level g+1, and G is a term which will be discussed below. Note that because the error calculation is based on the maximum allowable error, only the largest error term (for a particular wavelength band) resulting from Eq.(12) between each increment in surface discretization is important. When the largest error calculated using Eq.(12) for an increment on a surface discretization is less than the maximum allowable error, the error has converged, and the desired discretization level is known.

The term G in Eq. (12) is the estimated average surface irradiation for the environment -- referred to as the ambient term by Cohen et al. , 1988. G is used because different environments have different illumination levels. While a radiosity change of .001 units might be significant in a dim environment, this radiosity change might be insignificant in a bright environment. Therefore, a reference point based on the level of illumination is needed Furthermore, the reference level needs to be global, rather than based on the radiosity of the individual surface. A change of .001 units would be large percentage change for a surface with current radiosity of .001. However , if the average radiosity in the scene is 1. unit, the .001 change in radiosity of the surface will not be perceptible in the final image. G is given by:

$$G = (\Sigma E_k F_{*k}) / (1 - \rho_{ave}) \qquad (13)$$

where ρ_{ave} is the area averaged reflectivity of the environment, and the summation is overall all surfaces k, with F_{*k} being approximate form factors given by:

$$F_{*k} = A_k / \Sigma A_l \qquad (14)$$

G is calculated only once at the beginning of program execution.

Since the discretization of directional surfaces does not appear in the final image, the discretion does not have to be saved after energy is shot from the directional surface, and different

discretizations can be used each time the surface shoots, resulting in computational savings. The first time a directional surface shoots, the distribution of the incident radiance will be highly directional, since there will be a very high incident radiance from a relatively small solid angle in the direction of the light source and relatively small incident radiance from the rest of the environment. A highly directional incident distribution requires a high surface discretization to model reflection from the surface. For subsequent shots, the incident distribution will be much more uniform. Even for a mirror-like surface, if the incident distribution is nearly uniform, the reflected distribution is nearly uniform. As a result, for subsequent shots, a much cruder discretization of the directional surface is required.

Figures 3 and 4 show the results of using adaptive subdivision. These figures were produced by using a ray tracing pass following the radiosity solution to render the specular surface.

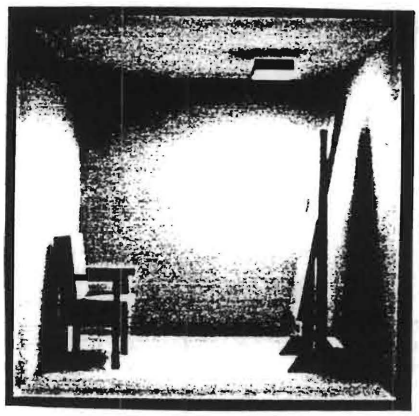


Figure 3 - Environment with 751 surfaces showing the effect of specular reflection.

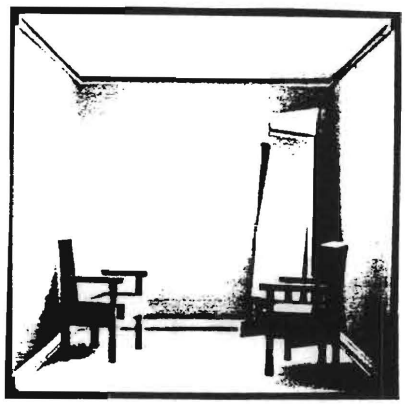


Figure 4 - Environment with 1459 surfaces showing the effect of specular reflection.

In Fig. 3, the scene is divided into 751 patches, with one large specular surface. In the radiosity solution energy was shot from the specular surface three times before the solution converged. The

initial discretization of the specular surface was 3x1. The first time energy was shot from the specular surface four iterations were required to reach the discretization at which the error term dropped below the prescribed tolerance -- resulting in a discretization of 12x4 (i.e. 48 patches). In the second and third shots from the specular surface only two iterations were required to reach the appropriate discretization.

In Fig. 4, the scene is divided into 1459 patches, with the same large specular surface. Again energy was shot from the specular surface three times before the solution converged. In this case, only two iterations were required to determine the specular discretization for all three shots. This is primarily because the walls are relatively bright, making the relative effect of the specular surface reflection smaller.

In general, convergence to the correct discretization is not monotonic. For example, a surface may not be visible from a specular surface when a crude surface discretization is used, but may become visible when the discretization is refined. At least the first time energy is shot from a specular surface, the maximum error value can be expected to increase for a few iterations before decreasing and converging.

5. General, Non-Lambertian, Non-Mirrorlike Surfaces

By examining Eqs. (7), (9), and (10) it can be seen that the complexity of the calculations for shooting from a patch is greater for a general directional surface than for a Lambertian surface or a mirror-like surface. For both Lambertian and mirror-like surfaces, the radiance to be shot out through a given pixel is given by a single value, while a summation over all pixels is required to compute the value for a general surface. Intuitively, the work required to distribute energy from a general surface should not be substantially greater than that required for a Lambertian or mirror-like surface.

As discussed by Shao et al., for near mirror-like surfaces, in which the reflected energy is concentrated in a small solid angle, the work required in the summation can be reduced by summing over a small number of hemi-cube pixels q around the pixel p' in the mirror direction from surface p. For general surfaces however, which are neither Lambertian nor near mirror-like, the number of pixels from which significant amounts of energy are received is on the order of the number of pixels on the hemi-cube. To completely shoot energy from a general surface $O(N_{pixel}^2)$ operations are needed, rather than the $O(N_{pixel})$ required for Lambertian or near mirror-like surfaces.

Equation (9) for a general surface is inefficient because it doesn't take advantage of the low resolution needed to model the reflected energy distribution relative to the hemi-cube resolution needed to model the incident radiation. As shown in Fig. 5, the incident distribution has high radiance gradients, while the reflected distribution has smooth gradients . We will take advantage of the averaging effect of non-mirrorlike surfaces to reduce the number of calculations indicated by Eq. (9). We will do this by introducing the reflectance hemisphere and hemi-cube pixel groups.

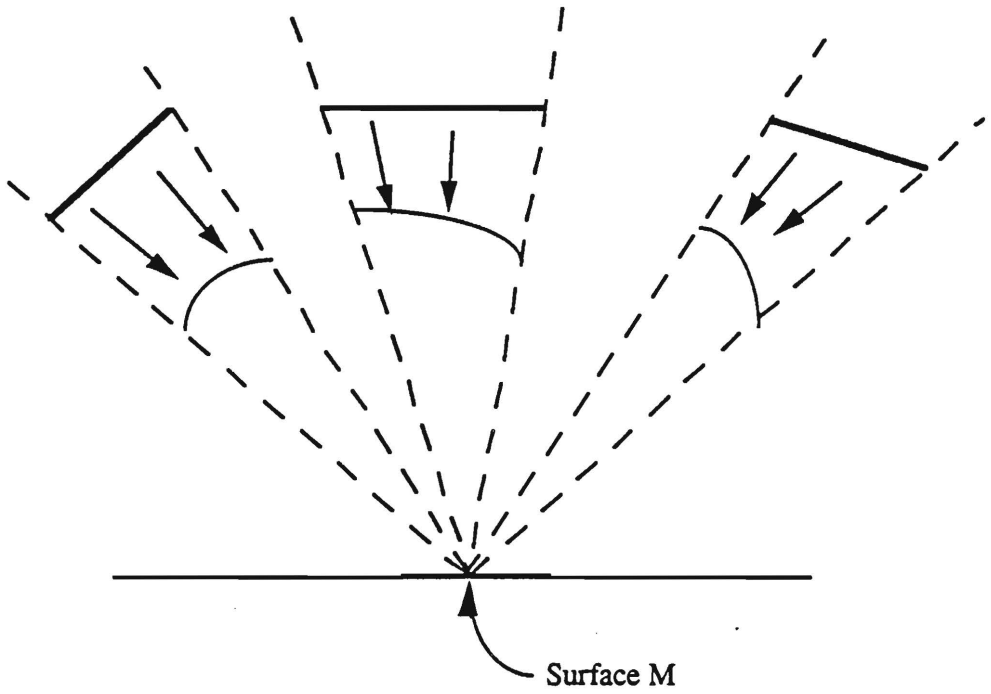


Figure 5a - Two dimensional view of hemi-cube showing how the incident distribution of energy striking surface M has sharp edges.

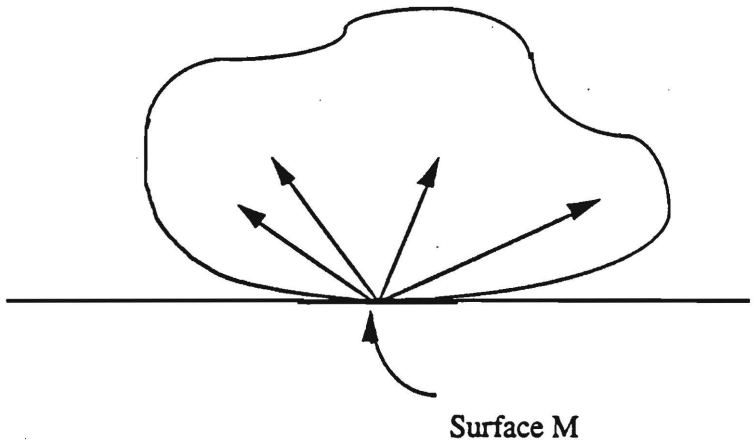


Figure 5b - Two dimensional view of the possible reflected distribution of energy leaving surface M. Note the relatively smooth shape of the distribution.

5.1 Reflectance Hemispheres

A reflectance hemisphere consists of a unit hemisphere divided into small elements, as shown in Fig. 6 . For a particular material, wavelength band and angle of incidence, each node on the reflectance hemisphere is assigned a reflectance value whose magnitude depends on the location of the node on the hemisphere surface. For general reflectors, the number of hemisphere nodes required is very small compared to the number of hemi-cube pixels. The bidirectional reflectance is modeled using the geometry of a hemisphere rather than a hemi-cube because it allows for uniform directional sampling.

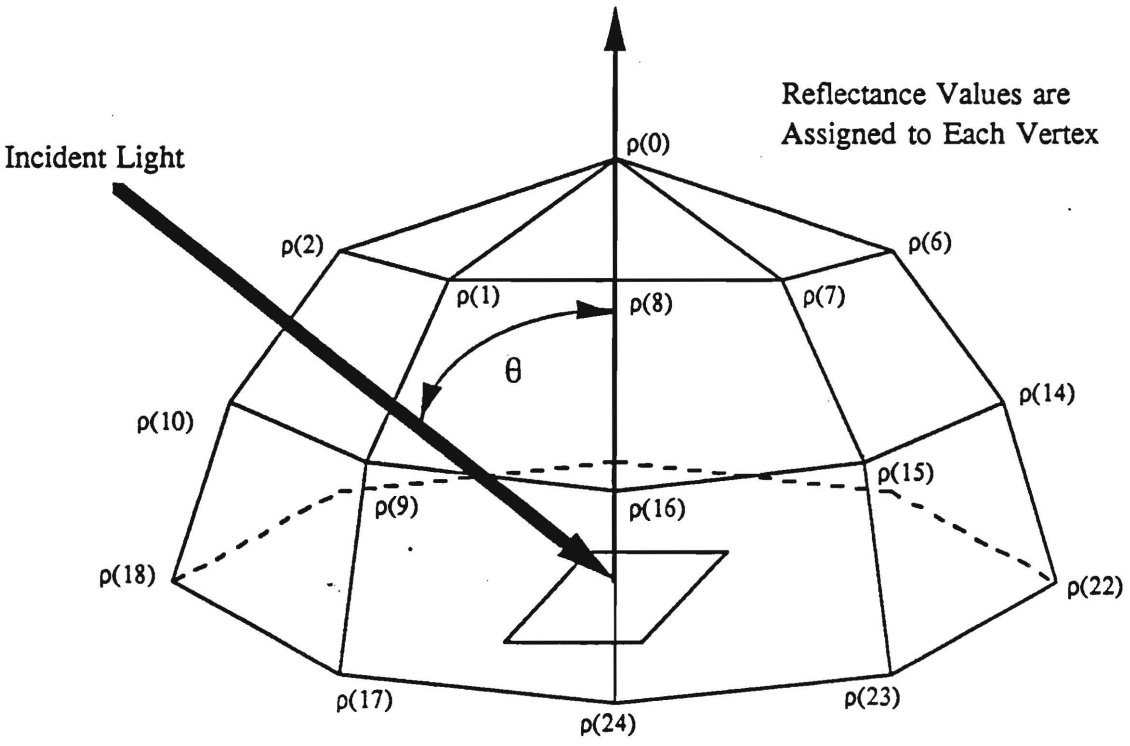


Figure 6 - Each reflectance hemisphere stores reflectances at each node for a particular angle of incidence.

Several different reflectance hemispheres are needed to store the reflectance characteristics of a material-- one hemisphere for each incidence angle and wavelength band considered. These reflectance hemispheres are generated from the bidirectional reflectance functions. During program execution when light is incident at an arbitrary angle from the normal, a new reflectance hemisphere is generated by interpolating between the values of the reflectance at the nodes of the two known reflectance hemispheres.

5.2 Hemi-cube Pixel Groups

Solving the hidden surface problem for calculating the magnitude of incident energy requires a relatively high hemi-cube pixel resolution even for Lambertian surfaces. For a Lambertian surface, however, the directional distribution of incident energy doesn't need to be estimated. For a general surface, the directional distribution needs to be estimated, but not at the high discretization level used for the hidden surface problem. We propose a cruder estimation of the incident energy by defining hemi-cube pixel groups, as shown in Fig. 7 . Consider a hemi-cube over a surface k. The energy incident per unit time and area of surface k for a pixel group g is

$$H_{K,g} = \sum I_{i,q} \Delta F_q \pi \tag{15}$$

where the sum is over all hemi-cube pixels q in group g, and the incident radiance is the unshot radiosity divided by π if the surface Q is Lambertian, and is $I_{u,Q,K}$ if Q is non-Lambertian.

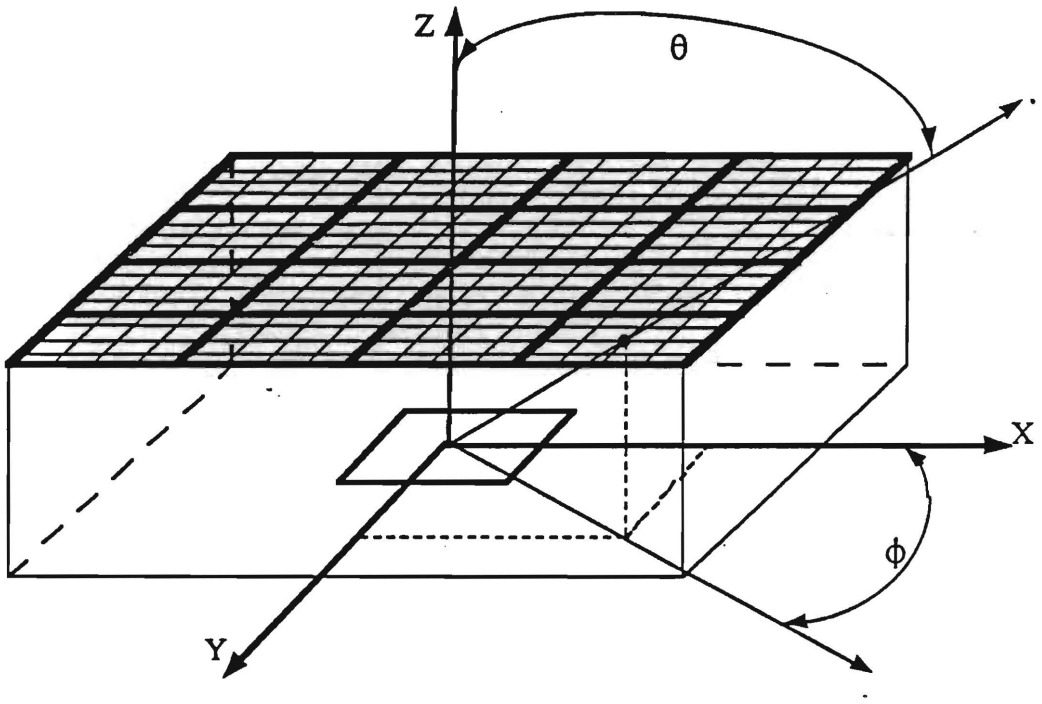


Figure 7 - Sketch showing hemi-cube pixel groups and the angles associated with them. Each group shown above contains 16 pixels. (Note: for clarity, only the pixel groups on the top hemi-cube face are shown.)

5.3 Calculating Delta Radiosities

The values from the reflectance hemispheres and the incident energy from hemi-cube pixel groups can be used to estimate the radiance to be shot out through each pixel. The reflected radiance at reflectance hemisphere node n is

$$I_n = \sum \rho_{bd}(g,n)\ H_{K,g} \qquad (16)$$

where $\rho_{bd}(g,n)$ is evaluated by interpolating between known hemisphere values to get values for an incident direction which passes through the center of hemi-cube pixel group g. Since the number of hemisphere nodes is relatively small compared to the number of hemi-cube pixels, this is a relatively short calculation. Given the value of I_n at each reflectance hemisphere node n, the value of I_p to be shot through each hemi-cube pixel p can be found by interpolation. Interpolated values can be estimated quickly by assigning the value I_n (divided by an appropriate scaling factor) to each vertex on the reflectance hemisphere as a color, the same way that surface id is used as a color when using a hardware z-buffer to calculate form factors. The surfaces of the hemisphere are projected onto the hemi-cube using Gouraud shading. The value of I_n can be read out of the color buffer after the projection. There is some error associated with using the Gouraud shading, but a good estimate can be obtained quickly using this method. A different set of I_p needs to be calculated for each wavelength band.

The increment in the radiosity of a surface P receiving energy from surface K through pixel p is:

$$\Delta B_P = \rho_{dh,P}(p)\ I_{u,K,p} \Delta F_p \pi A_K / A_P \qquad (17)$$

Using the reflectance hemisphere and hemicube pixel groups, the $O(N_{pixel}^2)$ operations to shoot out energy from a general surface is reduced to $O(N_{pixel})$ operations to calculate the hemi-cube pixel group contributions, $O(N_{reflectance\text{-}node}^2)$ operations to compute the reflectance hemisphere radiances, plus $O(N_{pixel})$ operations to compute the interpolated radiances.

Figures 8a-c are three images generated witht he improved radiosity method. They show the effect of reflectance of a test surface on the other surfaces in a scene. The top of the table is the test surface, and it has been assigned Lambertian, mirror-like, and general reflectance functions in Figs. 8a, b, and c respectively. (Data for the reflectance of typewriting paper from Siegel and Howell, 1981 was used for the general surface.) Red and blue spotlights are aimed at the surface, and all of the surfaces are gray. In all three images the scene is rendered with the radiosity solution only (before the second view dependent pass). In Fig. 8a, the effect of the Lambertian reflectance is to illuminate all surfaces with a purple light. In Fig. 8b, there are clear red and blue spots resulting from the mirror-like reflection. In Fig. 8c, there are slight concentrations of red and blue resulting from the directional reflectance of the surface.

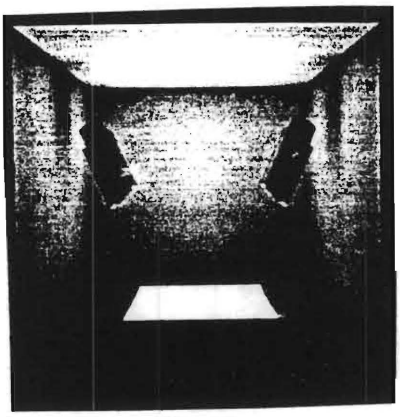


Figure 8a - Diffuse reflection from test surface.

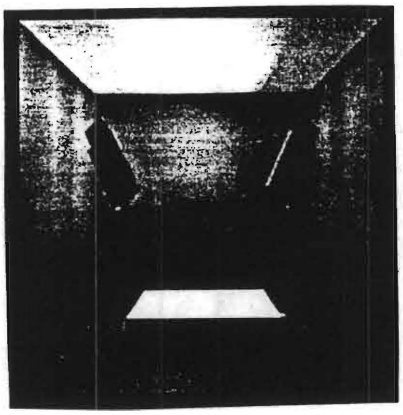


Figure 8b - Specular reflection from test surface.

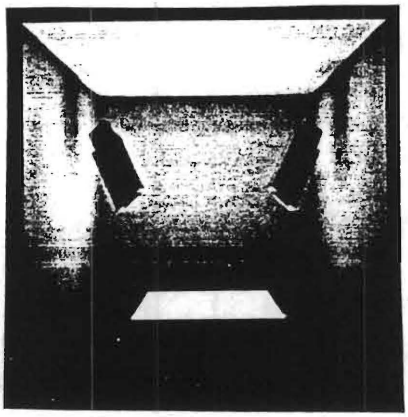


Figure 8c - Bidirectional reflection from test surface.

6. Summary

We have examined two issues arising with an explicit radiosity method for non-Lambertian surfaces -- meshing non-Lambertian surfaces, and efficiently calculating the effects of non-Lambertian, non-mirrorlike surfaces. To address the meshing issue we applied two general principles -- that meshing of non-Lambertian surfaces should depend on their effect on other surfaces, and that the effect on a surface should be measured relative to the global illumination of the scene. To address the issue of reflection from non-Lambertian surfaces, we took advantage of the fact that different discretizations of the hemisphere of directions above the surface can be used for calculating incident energy and for calculating reflected radiance.

7. Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant #ECS-8909251.

8. References

Baum, D. R., Rushmeier, H. E. and Winget, J. M. (1989) "Improving Radiosity Solutions Through the Use of Analytically Determined Form-Factors," *Computer Graphics* (SIGGRAPH '89 Proceedings) 23(3):325-334.

Baum, D.R., Mann, S., Smith, K.P., and Winget, J.M. (1991) "Making Radiosity Usable: Automatic Preprocessing and Meshing Techniques for the Generation of Accurate Radiosity Solutions," *Computer Graphics* (SIGGRAPH '91 Proceedings) 25(3):51-60.

Campbell, A.T. and Fussell, D.S. (1990) "Adaptive Mesh Generation for Global Diffuse Illumination," *Computer Graphics* (SIGGRAPH '90 Proceedings) 24(4):155-164.

Chen, H. and Wu, E.-H., (1990) "An Efficient Radiosity Solution for Bump Texture Generation," *Computer Graphics* (SIGGRAPH '90 Proceedings) 24(4):125-134.

Chen, S.E., Rushmeier, H. E., Miller, G. and Turner, D. (1991) "A Progressive Multipass Method for Global Illumination," *Computer Graphics* (SIGGRAPH '91 Proceedings) 25(3):165-174.

Cohen, M.F. and Greenberg, D.P. (1985) "The Hemi-cube: A Radiosity Solution for Complex Environments," *Computer Graphics* (SIGGRAPH '85 Proceedings) 19(3):31-40.

Cohen, M.F., Greenberg, D.P., Immel, D.S. and Brock, P.J. (1986) "An Efficient Radiosity Approach for Realistic Image Synthesis," *IEEE Computer Graphics and Applications* 6(2):26-35.

Cohen, M.F., Chen, S.E. , Wallace, J.R. and Greenberg, D.P. (1988) "A Progressive Refinement Approach to Fast Radiosity Image Generation," *Computer Graphics* (SIGGRAPH '88 Proceedings) 22(4):75-84.

Goral, C.M., Torrance, K.E., Greenberg, D.P. and Battaile, B.(1984) "Modeling the Interaction of Light Between Diffuse Surfaces" *Computer Graphics* (SIGGRAPH '84 Proceedings) 18(3):213 - 222.

Immel, D.S., Cohen, M.F. and Greenberg, D.P. (1986) "A Radiosity Method for Non-Diffuse Environments", *Computer Graphics* (SIGGRAPH '86 Proceedings) 20(4):133-142.

Kajiya, J. T. (1986) "The Rendering Equation," *Computer Graphics* (SIGGRAPH '86 Proceedings) 20(4):143-150.

Malley, T.J.V., (1988) "A Shading Method for Computer Generated Images," Master's Thesis, The University of Utah.

LaSaec, B. and Schlick, C. (1990) "A Progressive Ray-Tracing Based Radiosity with General Reflectance Functions," Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics (Rennes, France) 53-66.

Nishita, T. and Nakamae, E. (1985) "Continuous Tone Representation of 3-D Objects Taking Account of Shadows and Interreflection," *Computer Graphics* (SIGGRAPH '85 Proceedings) 19(3):23-30.

Rushmeier, H.E. and Torrance, K.E. (1990) " Extending the Radiosity Method to Include Specularly Reflecting and Translucent Materials", *ACM Transactions on Graphics* 9(1):1-27.

Shao, M.-Z., Peng, Q.-S., Liang, Y.-D. (1988) "A New Radiosity Approach by Procedural Refinements for Realistic Image Synthesis," *Computer Graphics* (SIGGRAPH '88 Proceedings), 22(3):93-101.

Siegel, R. and Howell, J.R. (1981) *Thermal Radiation Heat Transfer.* Hemisphere, Washington New York London.

Sillion, F. and Puech, C.(1989) "A General Two-Pass Method Integrating Specular and Diffuse Reflection" *Computer Graphics* (SIGGRAPH '89 Proceedings) 23(3):335-344.

Sillion, F., Arvo, J. R., Westin, S. H. and Greenberg, D. P.(1991) "A Global Illumination Solution for General Reflectance Distributions," *Computer Graphics* (SIGGRAPH '91 Proceedings) 25(3):187-196.

Wallace, J.R., Cohen, M.F. and Greenberg, D.P.(1987) "A Two Pass Solution to the Rendering Equation: A Synthesis of Ray Tracing and Radiosity Methods," *Computer Graphics* (SIGGRAPH '87 Proceedings) 21(4): 311-320.

A Progressive Multi-Pass Method for Global Illumination

Shenchang Eric Chen, Holly E. Rushmeier†, Gavin Miller, Douglass Turner

Advanced Technology Group
Apple Computer Inc.

†The George Woodruff School of Mechanical Engineering
Georgia Institute of Technology

ABSTRACT

A new progressive global illumination method is presented which produces approximate images quickly, and then continues to systematically produce more accurate images. The method combines the existing methods of progressive refinement radiosity, Monte Carlo path tracing and light ray tracing. The method does not place any limitation on surface properties such as ideal Lambertian or mirror-like. To increase efficiency and accuracy, the new concepts of light source reclassification, caustics reconstruction, Monte Carlo path tracing with a radiosity preprocess and an interruptible radiosity solution are introduced. The method presents the user with most useful information about the scene as early as possible by reorganizing the method into a radiosity pass, a high frequency refinement pass and a low frequency refinement pass. The implementation of the method is demonstrated, and sample images are presented.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation - Display Algorithms. I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism
General Terms: Algorithms
Additional Key Words and Phrases: Radiosity, Ray Tracing, Monte Carlo, Caustics, Global Illumination, Progressive Refinement.

INTRODUCTION

Generating realistic images of complex scenes is still far from a real time process. To handle this problem, Bergman et. al. [1] introduced the concept of "adaptive refinement" for generating high quality images. An adaptive refinement method has two fundamental properties: the image continues to improve indefinitely with time, and the most useful information is produced earliest in the rendering process. In this paper we present a global illumination method for generating physically accurate images which follows the adaptive refinement paradigm.

In Bergman's adaptive refinement method, only simple shading models are presented. The global illumination effects, such as shadowing and inter-reflection between surfaces, are ignored. In addition, the refinement process leaps from one shading model to another. The transition between refinement steps is not smooth. Cohen et. al. [2] subsequently extended the concept to develop a "progressive refinement" radiosity method that allows the user to view the images as the radiosity solution evolves. The new method uses a more sophisticated global illumination model and generates images that progress smoothly and gracefully to the final image. The idea of progressive refinement is readily applied to ray tracing methods as well. Painter, Sloan [3] and Ward [4] have presented ray tracing methods that evolve by casting increasing numbers of rays to increasing numbers of pixels.

Both pure radiosity and pure ray tracing solutions to the global illumination problem have disadvantages. Radiosity methods require careful, detailed meshing to correctly capture shadows, which may have very high spatial frequency [5], [6]. Radiosity methods also require an excessive computation time and storage to directly solve for non-diffuse reflections [7]. While radiosity methods have been developed with the capability of capturing caustic effects (e.g., [8]), meshing methods to guarantee the capture of these effects do not exist. Ray tracing methods can be formulated to produce physically accurate solutions (i.e., [9], [10]). However, such methods require huge numbers of rays to be cast per pixel to avoid perceptible noise in the image. While caustic effects can be captured with eye ray tracing, using "backward ray tracing" (i.e., light ray tracing) and "caustic maps" ([11], [12]) is generally more effective.

Because of the relative advantages and disadvantages of radiosity and ray tracing, many hybrid or "multi-pass" methods have been developed. The first such multi-pass method was the radiosity method developed by Nishita and Nakamae [13], in which different techniques were used for direct and indirect illumination. Wallace et. al. [14] and Sillion and Peuch [15] developed hybrid methods in which an extended radiosity method was followed by a ray tracing pass to solve for view dependent directional reflections. Shirley [16] developed the most extensive multi-pass method to date. In Shirley's method, radiosity is used for indirect illumination, Monte Carlo ray tracing is used for direct illumination, and light ray tracing is used for caustics.

While multi-pass methods such as Shirley's can produce excellent and physically accurate images, they may still produce noticeable artifacts. The surface discretization from the radiosity pass is still used in the final image. If the environment has strong indirect illumination, the quality of the final image will be strongly dependent on the meshing of the visible surfaces. Diffuse surfaces are still assumed to be ideal Lambertian in the final image. There is no mechanism for rendering surfaces which are neither strongly

ACM-0-89791-436-8/91/007/0165 $00.75

directional nor ideal Lambertian.

Another drawback of all the previous multi-pass methods is that they do not provide intermediate feedback to the user. Since global illumination rendering is generally a very lengthy process, intermediate feedback is very important in detecting mistakes early on . In the case of Shirley's method, the user must wait for many higher order diffuse interreflections which produce little new information about the scene in the progressive radiosity phase before sharp shadows, textures and surface bumps can be obtained in the Monte Carlo ray tracing phase. The ray tracing phase then progresses in the order of pixels. No complete image is available for fast feedback until every pixel is finished.

We present an extended, reorganized version of the multi-pass concept which is designed to overcome these disadvantages. The method consists of a series of passes which continuously provide user feedback. The rendering process begins with a progressive refinement radiosity pass with extended form factors computed by ray tracing for non-diffuse surfaces. This pass provides a good approximation to the overall illumination. A high frequency refinement pass follows to perform a Monte Carlo path tracing from the eye and the lights to create shadows and caustics. Unlike earlier methods, the path tracing is only directed at surfaces that are considered "bright" enough to create high frequency details. A caustics reconstruction technique is introduced to compute a caustic map for each surface. A low frequency refinement pass continues to refine the image using Monte Carlo path tracing for accurate low frequency illumination effects such as color bleeding. The low frequency refinement makes use of the results from the radiosity pass for high order reflections. Therefore, it should be faster than pure path tracing. Since the pass is performed pixel by pixel, the radiosity meshing artifacts are invisible in the final image.

In the new method, the user does not need to wait for the first pass to finish before the next pass begins. The radiosity pass can be interrupted to compute the high frequency details. All the three passes potentially can be run in parallel.

We begin with a description of the multi-pass method in the next section. We then describe how the method is organized into a progressive refinement solution, followed by a description of implementation and results. Conclusions and future directions are presented at the end.

EXTENDED MULTI-PASS METHOD

A solution for global illumination must account for all of the energy which can pass from sources of light to the eye. In the following sections, we present two ways of examining how our method solves the global illumination problem. Firstly, we examine how all possible light paths between the sources and the eye are accounted for. Secondly, we examine how all the terms in the rendering equations [9] are accounted for. We then present a detailed discussion of two new features of our method–light source reclassification and caustics reconstructions. Finally, we contrast the new extended method with existing multi-pass methods.

Accounting for All Light Paths

We use three techniques to find all the significant light paths:

1. Progressive Refinement Radiosity (PRR), with ray tracing for extended form factors.
2. Light Ray Tracing (LRT), which traces rays from light sources for caustic map generation.
3. Monte Carlo Path Tracing (MCPT), with distributed ray tracing [17] being a subset of MCPT.

We divide all possible light paths into four classes. Let *s* be a reflection or transmission off of a "specular-like" surface (i.e. highly directional but not necessarily a perfect mirror), and *d* be a reflection or transmission off of a "diffuse-like" surface (i.e. weakly direction but not necessarily Lambertian). Using s and d to denote reflection/transmission events, the four path classes are shown in Fig. 1.

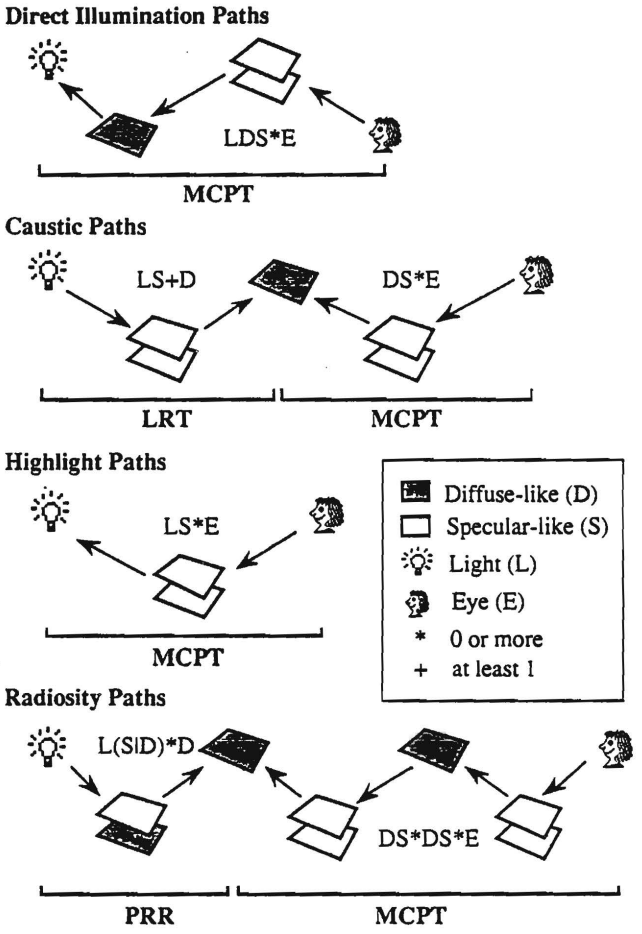


Fig. 1. Four classes of light paths are shown both pictorially and with Heckbert's style of notations[18]. The paths are followed with three techniques–PRR, LRT and MCPT. Arrows indicate the direction in which the path is followed.

Direct illumination paths refer to paths from sources to the eye via one *d* followed by zero or more *s*'s. These paths are followed using MCPT. As soon as a ray traced from the eye hits a *d*, another ray is cast at a light source. The direction of this second ray is chosen using a probability density function (*pdf*) based on the area distribution of energy/time on the light sources. An example of these paths is the shadowing effects created by light sources.

Caustic paths contain one or more *s*'s, a single *d*, and zero or more *s*'s before the eye. These paths are followed using LRT and the results are deposited on the caustic maps attached to diffuse surfaces. The caustic maps are created from the intersections of the caustic paths with the diffuse surfaces using a reconstruction method that will be described in "Caustic Reconstruction".

Highlight paths consist of zero or more *s*'s and are traced by MCPT. Highlight paths account for the direct rendering of light sources, the rendering of light sources through specular-like transmitters, and the production of specular-like highlights in opaque surfaces.

Radiosity paths contain at least two *d*'s. Radiosity paths are followed using a combination of PRR and MCPT. Paths from the eye up to the second *d* are followed using MCPT. Paths which continue on through any number of *s* and *d* until reaching the source are followed using PRR. Radiosity paths produce the classic radiosity effects such as color bleeding.

These four paths encompass paths with any combination of *s* and *d* between the source and the eye. This can be shown with the number of *d*'s contained in the paths. Any path that has at least two *d*'s is a radiosity path. Paths with one *d* is either a caustic path or a direct illumination path. Paths contain zero *d* belong to the highlight path.

Solving the Rendering Equation

In this section we describe how the strategies outlined above produce a solution to the rendering equation. The quantity of light which we compute at each step in the solution is the radiance, I, the light energy per unit time, projected area and solid angle (also called the intensity). The reflectance of each surface is given by its bidirectional reflectance, $\rho_{bd}(\theta_i,\phi_i;\theta_r,\phi_r)$, which is the radiance reflected in a direction *r* as the result of incident energy per unit time area and solid angle from a direction *i*.

As outline in the previous section, our method combines PRR, LRT and MCPT. MCPT involves using Monte Carlo methods to estimate integrals of various forms. The methods for estimating integrals are well established. To simplify the discussion in this section, many of the details of the Monte Carlo estimates will be omitted. These details can be found elsewhere, such as in [19].

For a particular view, an image is formed by finding an approximate solution to the following rendering equation for each pixel:

$$I_{pixel} = \int_{pixel\_area} I_o(p,\theta_r,\phi_r) f(x_s,y_s)\,dx_s\,dy_s \qquad \text{(eq. 1)}$$

where I_{pixel} is the pixel radiance, $I_o(p,\theta_r,\phi_r)$ is the radiance leaving a point "p" in the scene visible through screen location (x_S,y_S) in direction (θ_r,ϕ_r) to the eye, and $f(x_S,y_S)$ is a filtering function for anti-aliasing. I_O is a function of wavelength, and RGB values must be determined for I_{pixel}. To simplify discussion, we omit explicit wavelength dependencies and the transformations which convert a discrete wavelength sampling I_O to RGB values for I_{pixel}.

Pixel radiance is computed by averaging the results of many trial estimates of I_{pixel}. Each trial begins by tracing a ray from the eye through the pixel using a pdf based on $f(x_S,y_S)$. $I_o(p,\theta_r,\phi_r)$ must be found for the surface which the ray hits. The radiance leaving a surface as the sum of the emitted and reflected radiance:

$$I_o(p,\theta_r,\phi_r) = \underbrace{I_e(p,\theta_r,\phi_r)}_{\text{emitted}} + \underbrace{I_r(p,\theta_r,\phi_r)}_{\text{reflected}} \qquad \text{(eq. 2)}$$

In general, there is a transmitted radiance as well. However, since it is treated exactly analogously to the reflected component, we will omit transmission for now.

$I_e(p,\theta_r,\phi_r)$ must be specified, and $I_r(p,\theta_r,\phi_r)$ is given by:

$$I_r(p,\theta_r,\phi_r) = \int_{\cap} \rho_{bd}(\theta_i,\phi_i;\theta_r,\phi_r) I_i(\theta_i,\phi_i)\cos\theta_i\,d\omega_i \qquad \text{(eq. 4)}$$

where $I_i(\theta_i,\phi_i)$ is the radiance incident from a direction i, θ_i is the angle between the surface normal and the direction i, $d\omega_i$ is a differential solid angle, and the integral is over the incident hemisphere.

Formally, we can rewrite $\rho_{bd}(\theta_i,\phi_i;\theta_r,\phi_r)$ in terms of a diffuse-like component $\rho_l(\theta_i,\phi_i;\theta_r,\phi_r)$, which has a weak dependence on direction, and a specular-like component $\rho_h(\theta_i,\phi_i;\theta_r,\phi_r)$, which has a strong dependence on direction:

$$\rho_{bd}(\theta_i,\phi_i;\theta_r,\phi_r) = \rho_l(\theta_i,\phi_i;\theta_r,\phi_r) + \rho_h(\theta_i,\phi_i;\theta_r,\phi_r) \qquad \text{(eq. 5)}$$

We use ρ_l and ρ_h rather than ρ_d and ρ_s to avoid confusion with idealized Lambertian $(\rho_{bd} = \rho_d/\pi)$ and mirror-like $(\rho_{bd} = \rho_s/\cos\theta d\omega)$ reflectances. Lambertian and mirror-like reflectances may be included in ρ_l and ρ_h, but this decomposition does not require assuming these idealized reflectances.

We can express $I_r(p,\theta_r,\phi_r)$ as the sum of $I_h(p,\theta_r,\phi_r)$ and $I_l(p,\theta_r,\phi_r)$, where:

$$I_h(p,\theta_r,\phi_r) = \int_{\cap} \rho_h(\theta_i,\phi_i;\theta_r,\phi_r) I_i(\theta_i,\phi_i)\cos\theta_i\,d\omega_i \qquad \text{(eq. 6)}$$

$$I_l(p,\theta_r,\phi_r) = \int_{\cap} \rho_l(\theta_i,\phi_i;\theta_r,\phi_r) I_i(\theta_i,\phi_i)\cos\theta_i\,d\omega_i \qquad \text{(eq. 7)}$$

I_h is the radiance reflected from the specular-like component and I_l is the radiance reflected from the diffuse-like component. I_h is evaluated by MCPT. A direction is chosen using a pdf based on the reflectance, and the surface visible in that direction is found by ray casting. This process is performed recursively until a light source or a diffuse-like surface is encountered. The integral for I_h is then approximated using I_l of the last surface.

I_l is evaluated by decomposing it into four parts:

$$I_l = I_{l,s} + I_{l,c} + I_{l,h} + I_{l,l} \qquad \text{(eq. 8)}$$

where $I_{l,s}$ is light directly from light sources, $I_{l,c}$ is light from light sources via a series of specular-like reflections, $I_{l,h}$ is light from non-light sources via a a series of specular-like reflections, and $I_{l,l}$ is light from other diffuse-like surfaces. These four parts are expressed in the following equation:

$$I_{l,\beta}(p,\theta_r,\phi_r) = \int_{\cap} \rho_l(\theta_i,\phi_i;\theta_r,\phi_r) I_{i,\beta}(\theta_i,\phi_i)\cos\theta_i\,d\omega_i \qquad \text{(eq. 9)}$$

where $\beta = s,c,h,l$

All of the values on the right hand side for $I_{l,s}$ are known. The integral for $I_{l,s}$ is reexpressed as a sum of area integrals over each source g:

$$I_{l,s} = \sum_g \int_{A_g} \rho_l(\theta_i,\phi_i;\theta_r,\phi_r) I_g(\theta_g,\phi_g)\cos\theta_i\cos\theta_g V\,dA_g/r^2 \qquad \text{(eq. 10)}$$

where A_g is the area of light source g, θ_g is the angle between the normal to surface dA_g and the direction to p (i.e. the point where radiance is being evaluated) and r is the distance from dA_g to p. The integral over a source is estimated by casting a ray at a random point on the source. The term V is one if the light is visible in the direction, and is zero otherwise. The integral is then

approximated using the value of the integrand in that direction.

$I_{l,c}$ is found by LRT and is stored in a caustic map. When a ray from the eye hits a diffuse-like surface, the value of $I_{l,c}$ is computed from the map and then added to the radiance for that ray.

$I_{l,h}$ is evaluated recursively just as I_h is, with the exception that paths leading to the light source via a series of specular-like reflections are excluded to prevent double counting the caustic paths.

$I_{l,l}$ is evaluated as an integral in which the whole right hand side is known, by using the values of I_l calculated from the PRR solution to approximate the values of $I_{l,l}$. A ray is cast into a random direction in the incident hemisphere to determine the direction for evaluating the integrand. To avoid double counting the direct illumination paths, surfaces which are treated as light sources are not included in this integral.

The techniques described above are used to calculate trial values for each pixel. Let $\rho_{l,ave}$ and $\rho_{h,ave}$ be average reflectances. In an individual trial I_r is estimated using Russian Roulette [20] by rewriting I_r in the equivalent form:

$$I_r = \rho_{l,ave}(I_l/\rho_{l,ave}) + \rho_{h,ave}(I_h/\rho_{h,ave}) + (1-\rho_{l,ave}-\rho_{h,ave})0 \qquad \text{(eq. 11)}$$

Based on a choice of a uniformly distributed random number, I_r is estimated as either $I_l/\rho_{l,ave}$, $I_h/\rho_{h,ave}$ or zero. I_h is estimated by recursion. I_l is estimated by finding a trial value of $I_{l,s}$, adding $I_{l,c}$ and using Russian Roulette again to either estimate $I_{l,l}$ or $I_{l,h}$. The number of trials required depends on the value of the sample standard deviation of the estimated I_{pixel} compared to a user selected level of accuracy (i.e., [21], [22]).

Light Source Reclassification

Like many other global illumination methods, much more work is done in our method to estimate the radiances from light sources than from other surfaces. This is justified by the substantially greater radiosity of lights. Usually all self-emitting surfaces are defined as sources. In many environments, however, some non-emitting surfaces reflect enough energy to warrant treatment as light sources. Conversely, some self-emitters are very dim, and special treatment of these is not necessary. Goral et. al. [23] have treated surfaces directly illuminated by point lights as emitters. However, their motivation is to handle point lights rather than to capture strong indirect illumination.

Light source classification is performed after the PRR pass. A surface is classified as a light source if its radiosity, computed in the PRR pass, is considered large enough. Some self-emitters may not be considered as light sources and will be treated like the other reflecting surfaces.

Caustics Reconstruction

The caustic maps are constructed in the following steps:

First, the caustic map resolutions are determined for each surface either from a particular view or from the area of the surface if view independent solutions are desired. To compute view dependent resolutions, the scene is ray traced from the eye. The smallest ray-surface intersection kernel required for each surface by this pass determines the resolution of the caustic map required for that surface. This approach creates uniform maps instead of hierarchical ones like [18]. This is convenient when using MIP maps [24] or summed area tables [25] for caustic map anti-aliasing in the final pass. The algorithm will be conservative in that it will create more detailed caustic maps than required for objects which occupy both the foreground and the background. However, it will always provide adequate resolution for the near-by parts of a surface.

Second, for each light in turn, rays are fired off towards the specular surfaces. This may be achieved either by Monte Carlo sampling over a hemisphere, or by using one or more hemi-cubes [5] for item-buffer preprocessing [26]. Each ray is generated such that it carries the same amount of energy. The rays are reflected or refracted when they hit specular-like surfaces and stop when they encounter diffuse-like surfaces. All the decisions of treating a surface as specular-like or diffuse-like are made using Russian Roulette discussed previously. The result of this pass is that each diffuse-like surface has a list of ray-surface intersections for caustic rays.

Third, for each surface, the list of intersections is used to reconstruct a smooth caustic map. In the method proposed by Arvo [11], each ray deposits energy over four pixels with a bilinear ramp of the intensity. This is equivalent to depositing a square convolution kernel one pixel wide into the caustic map. The energy of a ray will be deposited into a rectangle which is d_u wide and d_v high in the parameter space where d_u is $1 / n_u$ and d_v is $1 / n_v$, and n_u and n_v are the resolutions of the caustic map in the u and v directions respectively. Since the caustic map is scaled by the tangent vector magnitudes when it is transformed to the world space during rendering, it is necessary to scale the intensity of the kernel in the caustic map (b) so that it will correspond to the ray energy (E) in the world space. For the sake of simplicity we assume that the tangent vectors are constant over the extent of the kernel.

A kernel with intensity b in the parameter space will correspond to an energy E in the world space according to the following equation.

$$b = E \,/\, \left(d_u d_v |p_u \times p_v|\right) \qquad \text{(eq. 12)}$$

where p_u is the u-tangent vector for the surface at the ray-surface intersection and p_v is the v- tangent vector.

Equation (eq. 12) is used to find b, which is deposited onto the final reconstructed caustic map.

For regions of the caustic map in which there are less than about four overlapping kernels per caustic map pixel, the resultant image will be very noisy (i.e., some pixels may not be hit by any ray at all). One method to diminish the noise is to use a larger convolution kernel for light deposition. This is called the fixed kernel method for the reconstruction of pdf's [27]. It has the disadvantage that it is hard to set the kernel size in a way which filters out the noise in sparse regions of the map and keeps high frequency detail in dense regions. Because of this limitation, adaptive kernel size methods should be used in which the size of the deposition kernel depends on the local density. This becomes a chicken and egg problem, in that the kernel size depends on the density, which in turn is the very thing we are trying to compute.

In this paper, a "nearest neighbors" method is used [27]. This involves expanding each convolution kernel until it covers n neighbors. A preprocessing step, for the sake of computational efficiency, is used to produce an "accumulation" image, at the reconstruction resolution, in which one pixel wide convolution

kernels are deposited for each ray (i.e., these kernels all have the amplitude of one). Then, around each ray-intersection, a rectangular region is expanded until it covers n neighbors, with n being computed by integrating the image over the rectangle. The ratio of the width to height of the rectangular region is set to the ratio of the corresponding tangent vector magnitudes. This means that the rectangle in the parameter space maps to an approximately square region in the world space. The dimensions of the rectangular region are then used to scale an elliptical-conical kernel which is deposited onto the final caustic map. The kernel amplitude is first set to normalize the filter and then multiplied by the intensity computed using eq. 12 to take into account the effects of the tangent vector magnitudes. This algorithm is illustrated in Fig. 2.

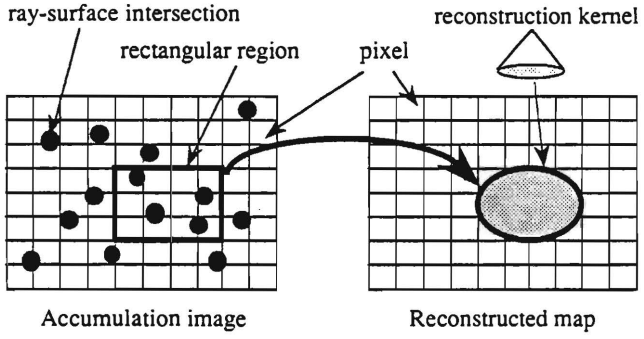


Fig. 2. Caustic map reconstruction

The integral of the accumulation image over the rectangular region may either be evaluated by direct summation or it could be computed using a summed-area table [25]. This would allow a binary search to be made on the rectangle size and would speed up the reconstruction process when ray intersections are very sparse. However, the subsequent large convolution kernels would still need to be deposited into the reconstructed map. When there are half the number of ray-surface intersections, the convolution kernels have twice the area. The computation time for the reconstruction depends linearly on the number of caustic map pixels, and is relatively insensitive to the number of ray-surface intersections.

Finally, the results are read out from the caustic maps during the Monte Carlo path tracing step as described previously.

Comparison to Previous Formulations

Previous global illumination methods can be described in the terms used in "Solving the Rendering Equation." In Wallace's method[14], I_r is decomposed into I_l and I_h, and ρ_l is Lambertian. I_l is given by the radiosity solution and I_h by distributed ray tracing. In Shirley's method[16], I_l is decomposed into $I_{l,s}$, $I_{l,c}$ and $I_{l,l+h}$. $I_{l,s}$ is evaluated as in our method (except that only self-emitters are sources). $I_{l,c}$ is calculated by interpolating a caustic map, rather than by reconstruction. $I_{l,l+h}$ is taken directly from the radiosity solution. Our reclassification of light sources allows improved shadows and caustics cast by indirect sources. The reconstruction produces a better representation of caustics. Finding $I_{l,l+h}$ by Monte Carlo integration, rather than using the PRR solution directly has two advantages. First, in the final rendering, the true bidirectional reflectance can be used. Second, the surface discretization used in the radiosity solution never appears in the final image, reducing the work required in meshing. Kajiya's pure MCPT approach is accurate but inefficient. Several researchers(e.g.[18]) have discussed the advantages of using LRT for caustics. Our major improvements are the caustic reconstruction and the use of PRR for higher order interreflections. Using the PRR solution has two advantages. The length of each individual trial is reduced, since paths end at diffuse-like surfaces. Also, the variance in the trials is reduced, reducing the number of trials for a particular level of accuracy. This variance reduction results from the value of $I_{i,l}$ being known, rather than being a high variance quantity which itself must be evaluated by recursion. Ward [10] used a similar strategy to reduce path length and variance by using "cached" radiance values for higher order interreflections.

PROGRESSIVE REFINEMENT

The method described in the previous section is not organized to present the user with the most important information at the earliest time possible. In this section we present a reorganization of the extended multi-pass method into a true progressive refinement method. The key ideas in this reorganization are an interruptible PRR, a high frequency refinement and a low frequency refinement pass.

First of all, the information to be presented must be prioritized. We have chosen the following ordering:

Overall global illumination of the environment:
- Approximate direct illumination
- Approximate diffuse-like interreflections
- Approximate specular-like reflections
- Approximate caustics

High spatial frequency variations in illumination:
- Sharp shadows
- Textures and surface bumps
- Specular-like reflections
- Caustics

Low spatial frequency variations in illumination:
- Accurate diffuse-like reflections

The user is presented with the overall global illumination using the first steps in a PRR solution. High spatial frequency variations are then presented by computing direct illumination, caustic and highlight paths. The low spatial frequency variations are calculated by computing the radiosity path.

Interruptible PRR

The first several iterations of a PRR solution provide a large quantity of useful information per unit time. However, as the method goes on, the rate at which images improve decreases dramatically. Each additional "shot" produces little visible effect on the solution. However, the solution must run for many more iterations to produce an accurate, converged solution. In our method an interrupted PRR solution is used to produce more detailed images, with the results of the completed PRR solution added in later.

An interrupted PRR solution can be used because of the linearity of the rendering equation. Let I_{int} be the interrupted PRR solution and I_{final} be the final PRR solution. Roughly, the process can be thought of as using the values of I_{int} as the radiosity values for the extended multi-pass method in one image, and the values of $(I_{final} - I_{int})$ in a second image, and then summing the results.

High Frequency Refinement

After the the PRR solution has been interrupted by the user, and a view chosen, high frequency refinement begins. Initially all surfaces are displayed with the values I_{int}. To approximate specular reflections quickly, a pass is made in which specular-like surfaces are ray traced from the eye (with the initial assumption that they are mirror-like). In this pass the interrupted PRR solution can also be modified using texture maps.

Next, the direct illumination, highlights and caustics resulting from each light source are calculated. To make the transition in the solution smooth, these effects are added into the image source by source. Three types of radiance values are calculated for each pixel – Iapprox, Itrue and Idisp. Iapprox is the value for the pixel found from PRR. Itrue is the value which has been accurately calculated for the pixel by MCPT. Idisp is the value displayed, and is the sum of Iapprox and Itrue. In the pass for each source g, the portion of Iapprox for each diffuse-like surface due to direct illumination and caustics from g is subtracted out, and value of Itrue is increased by using the caustic map for g and by following direct illumination paths from g. In estimating direct illumination, the bump maps and texture maps for that surface can be used. For each specular-like surface, the values of Iapprox and Itrue are replaced using the values of Iapprox and Itrue for the diffuse-like surfaces visible through the surface. The true value of ρ_h is used for the surface to find the visible diffuse-like surface (i.e., Note that the diffuse-like surface may be visible through a chain of specular-like reflections.) Treating specular-like surfaces in this way insures that diffuse-like surfaces will be treated in the same way when seen through a specular-like surface as when seen directly.

Let Ie be the emission, I_{prr} be the radiosity result maintained for each surface, and let the initial value of I_{prr} be I_{int}. The following is simplified pseudo-code for adding in the effect of each source g:

```
HighFrequencyPass(g) {
  Shoot out "negative" light from g to remove the effects of
  direct illumination and caustics of g from all Iprr's;
  Build caustic maps by shooting out caustic rays from g;
  For each pixel p {
    converged = false;
    trial = 0;
    sum_approx = 0;
    sum_true = 0;
    While not converged {
      trial++;
      shoot ray at p using f(p) as weighting function to find
      surface_hit;
      GetRadiance(g, surface_hit , trial_Iapprox, trial_Itrue);
      sum_approx += trial_Iapprox;
      sum_true += trial_Itrue;
      new_Iapprox = sum_approx/trial;
      new_Itrue = sum_true/trial;
      trial_Idisp =  Itrue(p) + new_Iapprox + new_Itrue;
      If ((standard_deviation( new_Itrue)/trial_Idisp)
        < accuracy) {
        converged = true;
        Itrue(p) += new_Itrue;
        Idisp(p) = trial_Idisp;
      }
    }
  }
}
```

```
GetRadiance(g, surface_hit,trial_Iapprox, trial_Itrue) {
  If Il chosen by Russian Roulette {
    trial_Iapprox = Iprr(surface_hit) ;
    trial_Itrue = estimate of Il,s† obtained by shooting at
    source g;
    trial_Itrue += estimate of Il,c† from caustic map of
    surface_hit;
  } Else If Ih chosen {
    Shoot ray in direction given by pdf based on ρh;
    GetRadiance(next_surface_hit , trial_I'approx,
    trial_I'true);
    trial_Itrue = trial_I'true†;
    trial_Iapprox = trial_I'approx†;
  }
}
```

For ease of explanation several details have been omitted from the pseudo-code. For example, more variables need to be saved to check the convergence of the value of radiance for each pixel. The convergence check does not need to be made after each trial, but after a group of trials. The number of trials in a group is determined on the fly, based on the initial estimates of the variance of the trials. The accuracy used in the convergence check must be smaller than the overall accuracy required for each pixel, because the errors from several passes will be summed. However, the effect of the higher accuracy requirement is mitigated by considering the sample standard deviation as a fraction of the total radiance for the pixel, not as the fraction of the current value of Itrue being estimated.

The quantities marked with a † need to be weighted to account for the Russian Roulette selection and multiplied by the surface reflectance. As noted by Arvo and Kirk, trees of rays, rather than strict paths may result in lower variances. Null results for rays directly to the eye increase the variance, so null selection in Russian Roulette is only used for higher order interreflections.

At the end of the high frequency refinement, the radiance of all direct illumination, caustic, and highlight paths have been accurately estimated. The user can either resume the interrupted PRR solution or can move on to the low frequency refinement pass.

Low Frequency Refinement

In the final pass, more accurate values for Iapprox are found by evaluating radiosity paths by MCPT with results from PRR. The value of I_{prr} for all light sources is set to Ifinal - Iint, since the direct and caustic contributions of Iint from these sources on other surfaces have already been calculated. The value of I_{prr} for all other surfaces is set to Ifinal since the effects of interreflections from these surfaces are now going to be estimated by integration over the hemisphere.

As in high frequency refinement, trials are made pixel by pixel until Itrue converges. Pseudo-code for calculating radiance in the low frequency refinement is given by:

```
/* hit_d indicates if a diffuse surface has been hit along the
  path. Initially, hit_d is false */
GetLRadiance(surface_hit, trial_Itrue, hit_d) {
  If Il chosen by Russian Roulette {
    If hit_d is false { /* hit the first d */
      Shoot ray out into hemisphere above surface_hit to find
      next_surface_hit;
      hit_d=true;
      GetLRadiance(next_surface_hit , trial_I'true, hit_d);
```

```
        trial_Itrue = trial_I'true†;
      } Else { /* hit the second d */
        trial_Itrue = Iprr(surface_hit);
      }
    } Else If Ih chosen {
      Shoot ray in direction given by a pdf based on ρh to find
      next_surface_hit;
      GetLRadiance(next_surface_hit , trial_I'true);
      trial_Itrue = trial_I'true†;
    }
  }
```

When solving pixel by pixel, the change in image quality per unit time is extremely slow. By storing extra data per pixel for convergence checks, the refinement can proceed by doing passes of one trial (or some small set of trials) per unconverged pixel, and displaying the intermediate results.

IMPLEMENTATION AND RESULTS

The multi-pass method has been implemented on a rendering testbed developed at Apple. A test scene was constructed to demonstrate the ideas presented in the paper. The scene contains a diffuse area light and two spotlights illuminating a diffuse room with a planar mirror, a mirrored-surface sphere, a glass sphere, a diffuse box and stick sculptures. The scene is designed to test illumination effects such as sharp shadows, soft shadows, color bleeding, caustics, specular reflection and refraction.

Fig. 3 shows a series of images generated in the progressive refinement process. Fig. 3a was generated at the end of the radiosity pass. The scene was tessellated to 114 patches and 4358 elements with patches and elements defined as in [28]. The radiosity pass was implemented with the hemi-cube algorithm [29]. The extended form-factors due to the specular reflection and refraction were computed with ray tracing. This image provides a good overall approximation but is lacking in high frequency details, such as sharp shadows and caustics. Since the radiosity pass is view independent, the user can inspect the scene from various views before continuing the refinement. Fig. 3b was generated at the end of the high frequency refinement pass, where the direct illumination from the two spot lights and the overhead area light was replaced with a more accurate, view dependent Monte Carlo solution. Notice the improved shadows and caustics on the floor. In this particular example, the high frequency details are created mostly by the lights and the radiosity pass provides a good approximation to the low frequency component. Therefore, this pass effectively renders the image close to its final form as shown in Fig. 3c. In Fig. 3c, the low frequency component is replaced by the Monte Carlo solution. Artifacts, such as the extraneous dark blob next to the stick's shadow on the left wall, from the radiosity meshing are therefore eliminated.

The Monte Carlo path tracing sent from 16 to 256 paths per pixel per light, based on the variance of each pixel. The number of paths per pixel per light is 25 on average in the high frequency pass and 250 per pixel in the low frequency pass. The image resolution is 540 by 300. Item buffer preprocessing was used to speed up the tracing of initial rays from the eye[26].

The timing data in Fig. 3 are based on computing the images on a Silicon Graphics' Iris 4D GTX. Since the actual computation time is machine and implementation dependent, the relative timing for each pass is a better indicator of the cost of the solution.

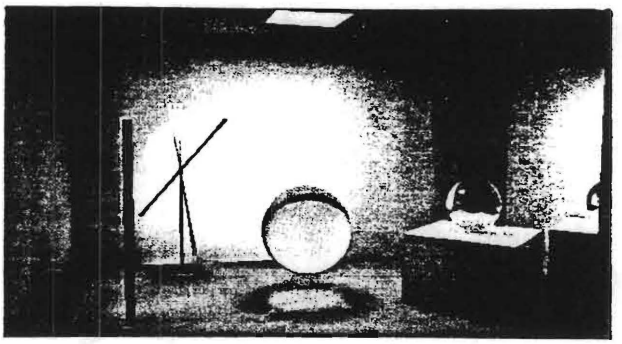


Fig. 3(a) At the end of the radiosity pass (12 minutes)

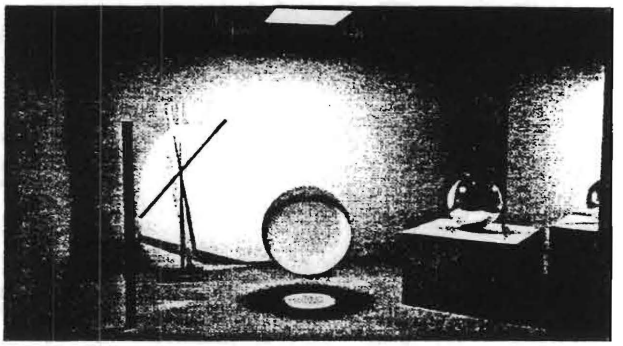


Fig. 3(b) At the end of the high frequency pass (4.5 hours)

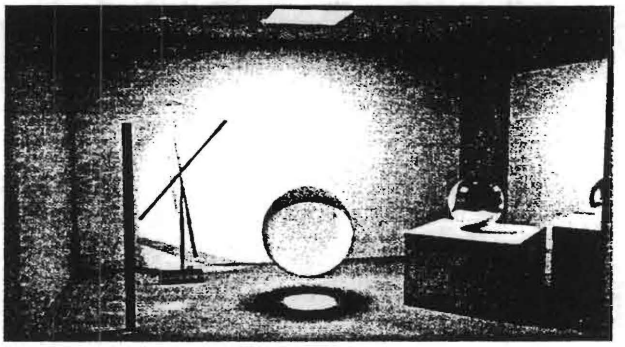


Fig. 3(c) At the end of the low frequency pass (21 hours)

Fig. 3. Images generated in the progressive refinement process. Notice the caustics on the floor created by the spot light pointing at the mirror in Fig. 3b and Fig. 3c.

Fig. 4 shows the difference between the images in Fig. 3 with color coding. The main difference between Fig. 3a, the radiosity image, and Fig. 3b, the high frequency refined image, is in the high frequency details. The difference between Fig. 3b and Fig. 3c is not very significant in this example.

Fig. 4(a): Fig. 3b - Fig. 3a

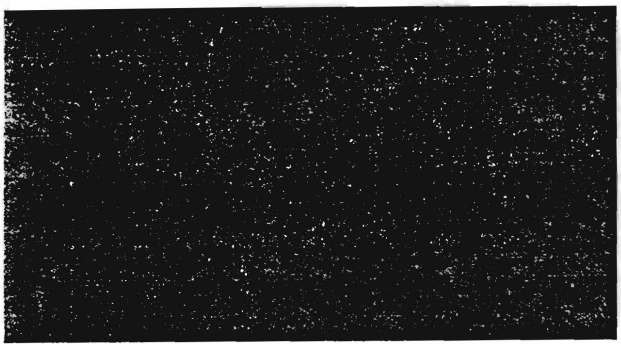


Fig. 4(b): Fig. 3c - Fig. 3b

Fig. 4. Differences of images in Fig. 3. Color coding is used to show the difference. Grey means no difference. Red means the first image's intensity is above the second one and conversely for blue.

The images in Fig. 3 are actually composed of partial images that show different illumination components. Fig. 3a is composed of Fig. 5a, the direct illumination from the light sources, and Fig. 5b, the interreflections between all the other surfaces. These two images were computed by recording the results of radiosity shootings from the light sources and non-light sources separately. Fig. 5a shows high frequency details that are not adequately represented with the coarse radiosity meshing. Fig. 5b is a better representation since the interreflection is very low frequency. However, some noticeable artifacts are still present. Fig. 5c is the direct illumination from the light sources computed by Monte Carlo path tracing. Fig. 5d shows the caustics computed by light ray tracing. Fig. 5b, Fig. 5c, Fig. 5d altogether compose the image in Fig. 3b, the results after the high frequency refinement. Fig. 5e shows the interreflections from non-light sources computed in the low frequency refinement pass. This image is more accurate than Fig. 5b but took significantly longer to compute. Fig. 5c, Fig. 5d and Fig. 5e altogether compose the image in Fig. 3c, the results after the low frequency refinement pass.

Fig. 6 shows the caustic maps before the reconstruction. The reconstructed results are shown in Fig. 5d. In this example, each light source sent out up to 262144 rays randomly based on the light's distribution function. The resolutions used for the caustic maps range from 4 to 128 pixels per side.

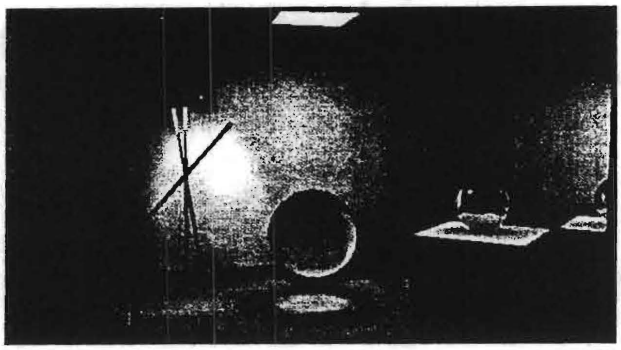


Fig. 5(a) Direct illumination and caustics computed by PRR.

Fig. 5(b) Interreflections computed by PRR.

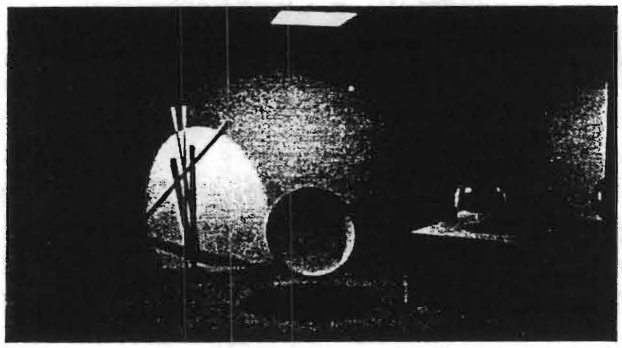


Fig. 5(c) Direct illumination computed by MCPT.

Fig. 5(d) Caustics computed by LRT.

Fig. 5(e) Interreflections computed by MCPT.

Fig. 5. Partial images which demonstrate different illumination components. These images compose the images in Fig. 3. Fig. 3a=Fig. 5a+Fig. 5b. Fig. 3b=Fig. 5b+Fig. 5c+Fig. 5d. Fig. 3c=Fig. 5c+Fig. 5d+Fig. 5e.

Fig. 6. Caustic maps before reconstruction. The light spots on the surfaces show the caustic maps with bilinear interpolation of ray-surface intersections. The reconstructed results using a variable sized kernel is shown in Fig. 5d[1].

Fig. 7 shows the same test scene computed with our method by omitting the high frequency refinement pass. Therefore, no ray was sent specifically to the directions of bright emitters or reflectors. This image was computed with 256 paths per pixel. The high level of noise indicates that high frequency sources should be treated separately from the low frequency ones.

Our method is also effective in creating shadows created by secondary light sources, such as a very bright diffuse reflectors. Fig. 8 shows a room in which the only lighting is from the spot light pointing at the diffuse wall. The rest of the room is lit by reflection from the wall, which acts as a secondary light source. The pole in the center casts a soft shadow on the floor because of this illumination. The shadow is well defined because in our method the bright wall surface is treated as a light source.

CONCLUSIONS AND FUTURE WORK

We have presented a new multi-pass progressive refinement method for global illumination. The method combines high efficiency and accuracy with practicality.

1. The gamma of Fig. 5d and Fig. 6 was increased for printing.

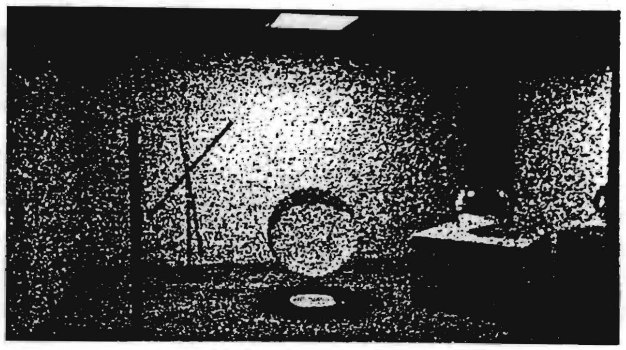


Fig. 7. Noisy image created by skipping the high frequency refinement pass. 256 paths were shot per pixel in the low frequency refinement pass.

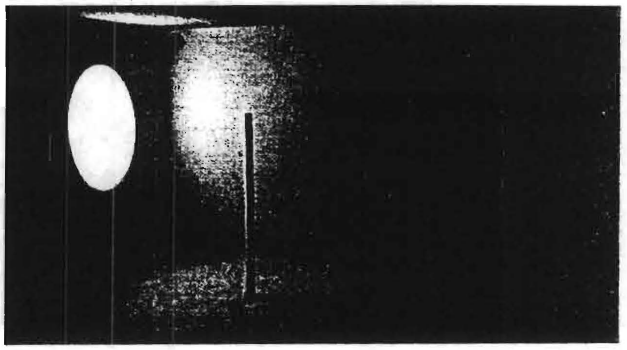


Fig. 8. Soft shadows created by indirect lighting. The shadow is well defined because the wall lit by the spot light is treated as a light source in the high frequency refinement pass. No low frequency refinement pass was performed on this image. The image took 2 hours to compute.

Efficiency and accuracy are obtained by using light source reclassification, caustics reconstruction and Monte Carlo integration with a radiosity preprocess. Light source reclassification allows the efficient approximation of the effect of strong indirect light sources. Caustics reconstruction provides a more accurate representation of caustic effects than simple interpolation. Monte Carlo integration with a radiosity preprocess eliminates radiosity discretization from the final image, while avoiding the high cost of pure path tracing.

The method has been made practical by reorganizing the solution into an interruptible progressive radiosity solution followed by high and low frequency refinement passes. Unlike previous radiosity methods, the user does not have to wait for multiple high order interreflections before seeing important high spatial frequency features such as sharp shadows and caustics. Unlike previous ray tracing methods, intermediate images show individual objects clearly, with a minimal level of noise.

Further areas of research include acceleration of the the method, improvements for handling a wide range geometric detail, and developing strategies for parallel implementation. The high frequency pass may be significantly accelerated with a preprocess to identify the number of objects each light can see. Different sampling strategies can be used to reduce the variance in the Monte Carlo integration. Radiosity methods are slow when a scene contains many small objects (e.g. plants with leaves, Venetian blinds, piles of rubber bands and paper clips). The trade-off between the illumination estimate provided by the initial radiosity

solution and the high spatial frequency variations caused by rendering large numbers of small objects needs to be examined. Parallel processing of the radiosity, high and low frequency refinement passes is straightforward. The potential of distributing the computation to a larger number of processors needs to be investigated.

ACKNOWLEDGEMENTS

We greatly appreciate Apple Computer for providing an exciting environment in which this research was conducted. The algorithm was implemented in a rendering testbed developed by Ken Turkowski, Douglass Turner and Shenchang Eric Chen. We would like to thank reviewers for their helpful suggestions, especially the 13-page hand-written comments from one of them. The second author would like to acknowledge the support in part by a grant from the National Science Foundation, ECS-8909251, "Progressive Refinement Algorithms for Radiant Transfer." Thanks also go to Robin Myers for his help in printing the color images.

REFERENCES

[1] Larry Bergman, Henry Fuchs, Eric Grant, Susan Spach, "Image Rendering by Adaptive Refinement," Computer Graphics (SIGGRAPH '86 Proceedings), V 20, N 4, Aug. 1986, 29-38.

[2] Michael Cohen, Shenchang Eric Chen, John R. Wallace, Donald P. Greenberg, "A Progressive Refinement Approach to Fast Radiosity Image Generation," Computer Graphics (SIGGRAPH '88 Proceedings), V 22, N 4, Aug. 1988, 75-84.

[3] James Painter and Kenneth Sloan, "Antialiased Ray Tracing by Adaptive Progressive Refinement," Computer Graphics (SIGGRAPH '89 Proceedings, V 23, N 3, July 1989, 281-288.

[4] Gregory J. Ward, "RADIANCE: A Tool for Computing Luminance and Synthetic Images," to appear in Lighting Design and Applications.

[5] Michael Cohen, Donald P. Greenberg "The Hemi-cube: A Radiosity Solution for Complex Environments," Computer Graphics (SIGGRAPH '85 Proceedings) V 19, N 3, July 1985, 31-40.

[6] A.T. Campbell,III, Donald S. Fussell, "Adaptive Mesh Generation for Global Diffuse Illumination," Computer Graphics (SIGGRAPH '90 Proceedings) V 24, N 4, August 1990, 155-164.

[7] David S. Immel, Michael F. Cohen, Donald P. Greenberg, "A Radiosity Method for Non-Diffuse Environments," Computer Graphics (SIGGRAPH '86 Proceedings), V 20, N 4, Aug. 1986, 133-142.

[8] Thomas J.V. Malley, *A Shading Method for Computer Generated Images*, Master's Thesis, University of Utah, June 1988.

[9] James T. Kajiya, "The Rendering Equation," Computer Graphics (SIGGRAPH '86 Proceedings), V 20, N 4, Aug. 1986, 143-150.

[10] Gregory J. Ward, Francis M. Rubinstein, Robert D. Clear, "A Ray Tracing Solution for Diffuse Interreflection," Computer Graphics (SIGGRAPH '88 Proceedings)," V 22, N 4, Aug. 1988, 85-92.

[11] James Arvo,"Backward Ray Tracing," SIGGRAPH '86 Developments in Ray Tracing seminar notes V 12, Aug. 1986.

[12] Mark Watt, "Light-Water Interaction using Backward Beam Tracing," Computer Graphics (SIGGRAPH '90 Proceedings), V 24, N 4, August 1990, 377-385.

[13] Tomoyuki Nishita, Eihachiro Nakamae, "Continuous Tone Representation of Three-Dimensional Objects Taking Account of Shadows and Interreflection," Computer Graphics (SIGGRAPH '85 Proceedings), V 19, N 3, July 1985, 23-30.

[14] John R. Wallace, Michael F. Cohen, Donald P. Greenberg, "A Two-Pass Solution to the Rendering Equation: A Synthesis of Ray Tracing and Radiosity Methods," Computer Graphics, (SIGGRAPH '87 Proceedings), V 21, N 4, July 1987, 311-320.

[15] Francois Sillion, Claude Puech, "A General Two-Pass Method Integrating Specular and Diffuse Reflection," Computer Graphics (SIGGRAPH '89 Proceedings), V 23, N 3, July 1989, 335-344.

[16] Peter Shirley, "A Ray Tracing Method for Illumination Calculation in Diffuse-Specular Scenes," Proceedings of Graphics Interface '90, May 1990, 205-212.

[17] Robert L. Cook, Thomas Porter and Loren Carpenter, "Distributed Ray Tracing," Computer Graphics (SIGGRAPH '84 Proceedings), V 18, N 3, July 1984, 137-144.

[18] Paul Heckbert, "Adaptive Radiosity Textures for Bidirectional Ray Tracing," Computer Graphics (SIGGRAPH '90 Proceedings), V 24, N 4, August 1990, 145-154.

[19] Holly Rushmeier, *Realistic Image Synthesis for Scenes with Radiatively Participating Media*, Doctoral Thesis, Cornell University, May 1988.

[20] James Arvo, David Kirk, "Particle Transport and Image Synthesis," Computer Graphics (SIGGRAPH '90 Proceedings) V 24, N 4, August 1990, 63-66.

[21] Werner Purgatohofer, "A Statistical Method for Adaptive Stochastic Sampling," Computers and Graphics, V 11, N 2, 157-162, 1987.

[22] Mark E. Lee, Richard A. Redner, Samuel P. Uselton, "Statistically Optimized Sampling for Distributed Ray Tracing," Computer Graphics (SIGGRAPH '85 Proceedings), V 19, N 3, July 1985, 61-68.

[23] Cindy M. Goral, Kenneth E. Torrance, Donald P.Greenberg and Bennett Battaile, "Modeling the Interaction of Light Between Diffuse Surfaces," Computer Graphics (SIGGRAPH '84 Proceedings), V 18, N 3, July 1984, 213-222.

[24] Lance Williams, "Pyramidal Parametrics," Computer Graphics (SIGGRAPH '83 Proceedings), V 17, N 3, July 1983, 1-9.

[25] Frank Crow, "Summed-Area Tables for Texture Mapping", Computer Graphics (SIGGRAPH '84 Proceedings), V 18, N 3, July 1984, 207-212.

[26] H. Weghorst, Gary Hooper, and Donald. P. Greenberg, "Improved Computational Methods for Ray Tracing," ACM Transactions on Graphics, V 3, N 1, January 1984, 52-69.

[27] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, ISBN 0 412 24620 1, 1986.

[28] Michael Cohen, Donald P. Greenberg, Dave S. Immel, Philip J. Brock, "An Efficient Radiosity Approach for Realistic Image Synthesis," IEEE Computer Graphics and Applications, V 6, N 3, March 1986, 26-35.

[29] Michael Cohen, Donald P. Greenberg "The Hemi-cube: A Radiosity Solution for Complex Environments," Computer Graphics (SIGGRAPH '85 Proceedings) V 19, N 3, July 1985, 31-40.

Geometric Simplification for Indirect Illumination Calculations

Holly Rushmeier Charles Patterson Aravindan Veerasamy

The Graphics, Visualization and Usability Center
Georgia Institute of Technology
Atlanta, GA 30332

Abstract

We present a new method for accelerating global illumination calculations in the generation of physically accurate images of geometrically complex environments. In the new method, the environment geometry is simplified by eliminating small isolated surfaces, and replacing clusters of small surfaces with simple, optically equivalent, boxes. A radiosity solution is performed on the simplified geometry The radiosity solution is then used in a multi-pass method to estimate the radiances responsible for indirect illumination. We present a preliminary implementation of the new method, and some initial images and timing results. The initial results indicate that using simplified geometries for indirect illumination calculations produces images in times significantly less than previous path tracing and multi-pass methods without a reduction in image quality.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation - Display Algorithms. I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

General Terms: Algorithms

Additional Key Words and Phrases: Geometric Simplification, Global Illumination, Monte Carlo, Progressive Refinement, Radiosity, Ray Tracing

Geometric Simplification for Indirect Illumination Calculations

1. Introduction

The outstanding problem in global illumination for computer graphics is the rendering of scenes containing very large numbers of geometric objects. While many useful global illumination methods have been proposed, the generation of physically accurate images of geometrically complex scenes still requires CPU hours on state of the art computer hardware. In this paper we present a method to accelerate global illumination calculations for complex scenes by geometric simplification for indirect illumination (GSII).This method is an extension of the progressive multi-pass method (PMM) described in Chen et al., 1991.

A fully correct solution for global illumination would require following all photons through a detailed geometric definition of the environment. Clearly, such a detailed solution is unnecessary to generate a realistic image. One approach to simplifying the calculations is to use many levels of geometric description (Kajiya, 1985).These levels include local lighting models (i.e. reflectances and transmittances), mappings (texture and bump maps) and object definitions. For example, in modelling the night sky, the moon can be adequately modelled by a texture map rather than by actually modelling the interaction of photons with the surface micro-structure of the moon's mountains and craters. Such approximations are intuitively acceptable, even though no rules governing their use have ever been developed.

Implicitly, many levels of object definition are used. In static images objects behind the viewer are modelled with relatively little geometric detail. On one hand this is viewed as "cheating," but is justifiable on the grounds that such details have very little impact on the overall scene illumination. In dynamic radiosity walk-throughs, generally only major objects are modelled for the sake of efficiency. Even without detail the user gets the overall impression of the illumination of the space. The goal of GSII is to begin to formalize the use of simplified and detailed geometries in rendering. Unlike the development of simplifications such as texture and bump mapping, a theoretical basis is developed for determining when GSII is appropriate.

We begin by briefly reviewing previous work for reducing the calculations required for global illumination of geometrically complex environments. Next we describe GSII and the theoretical basis for its application. Finally, an initial implementation of GSII is described, and preliminary timing results are presented.

2. Previous Work in Global Illumination of Complex Environments

A number of approaches for dealing with large numbers of geometric objects have been developed for the radiosity, ray tracing and hybrid approaches to global illumination.

In radiosity methods there are two levels of complexity -- the number of geometric objects K, and the number of subsurfaces N into which the K objects must be subdivided to capture illumination detail. In the original radiosity method (Goral et al. 1984, Nishita and Nakamae, 1985) $O(N^2)$ interactions had to be calculated to compute a solution. Cohen et al. (1986) developed the patch-element hierarchy to avoid the necessity of computing a large number of interactions between small subsurfaces, reducing the complexity of the calculation to O(KN). Hanrahan et al. (1991) built on the hierarchical subdivision concept to further reduce the calculation of surface interactions to a complexity of $O(K^2)$. While greatly reducing the complexity of radiosity calculations, hierarchical meshing of surfaces does not address the issue of how to deal with large numbers of objects (K) in the environment.

Xu et al., 1990, developed a technique of dividing space into V volume subdivisions, with an average of N/V subsurfaces in each volume. $O((N/V)^2)$ detailed interactions are computed within each volume. Less detailed interactions are computed between the subsurfaces and the other volumes. The complexity of the second set of calculations is difficult to assess since they depend on maintaining detailed directional geometric factors at volume boundaries which depend on the geometric complexity of the overall environment.

Ray tracing techniques for accurate global illumination, such as Kajiya's Monte Carlo path tracing (Kajiya, 1986), simplify the problem by only requiring an illumination calculation for objects within the field of view. Furthermore, the solutions for these objects only need to be calculated at screen resolution (e.g., individual radiances of a blades of grass need not be calculated if all of the blades project onto the same pixel.) For each of the P pixels, the illumination is found by following paths in D directions. Formally, the complexity of the calculation is just O(PD). Implicitly however, the number of objects K influences the calculations by influencing the number of directions D required for an accurate solution. More accurately the complexity then is O(PD(K)).

For both the radiosity and ray tracing approaches, the number of calculations is compounded by the average reflectance of the environment ρ_{ave}. In a progressive refinement radiosity solution (Cohen et al. 1988), the number of "shots" required increases with $1/(1-\rho_{ave})$, resulting in an $O((K^2/(1-\rho_{ave}))$ complexity. In a path tracing approach, the length of paths increase with $1/(1-\rho_{ave})$, resulting in a overall complexity of $O(PD(K)/(1-\rho_{ave}))$.

The goal of hybrid methods, such as the PMM, is to combine the advantages of the radiosity and ray tracing approaches. In a preliminary pass, a radiosity solution is performed on a coarse mesh to allow the user to quickly generate many different views. The coarse mesh gives a low constant for the $O(K^2)$ radiosity calculation. Since many views are used, the number of pixels P is very high and the constant for simply projecting the surfaces with precomputed radiosities is orders of magnitude smaller than the cost per pixel for a full path tracing solution. Detailed images of selected views are calculated with Monte Carlo path tracing using a radiosity preprocess. The preprocess is used to identify important secondary light sources, and so to reduce the number of directions required to a weaker function of K, D'(K). The preprocess is also used to estimate higher order reflections to eliminate the dependance on ρ_{ave}. The overall complexity of the path tracing pass then is just O(PD'(K)).

The approach used in this paper is to simplify the original K objects in the scene to an orders of magnitude smaller number S. The radiosity pass is then performed on this set of objects for a complexity of $O(S^2/(1-\rho_{ave}))$. In the Monte Carlo path tracing pass the indirect illumination is calculated using this solution as a preprocess, resulting in a complexity of O(PD'(S)).

3. Theory

In this section we present how GSII fits into a solution of the rendering equation, and how rules are developed to perform the geometric simplification

3.1 Solving the Rendering Equation

An image is formed by computing the radiance of each pixel based on the radiance of points in the environment viewed through that pixel. In the rendering equation (Kajiya, 1986), the radiance $I_o(p,\theta_r,\phi_r)$ leaving a point p in the environment in a direction specified in spherical coordinates as θ_r,ϕ_r is the sum of the emitted radiance at that point $I_e(p,\theta_r,\phi_r)$ and the reflected radiance $I_r(p,\theta_r,\phi_r)$. The emitted radiance must be specified to completely define a scene. The reflected radiance is calculated from the following integral:

$$I_r(p,\theta_r,\phi_r)= \int \rho_{bd}(p,\theta_r,\phi_r,\theta_i,\phi_i) I_i \cos\theta_i d\omega_i \qquad (1)$$

where I_i is the radiance incident from direction i, θ_i is the angle between the surface normal and direction i, $d\omega_i$ is a differential solid angle, and the integral is over the incident hemisphere. ρ_{bd} is the bidirectional reflectance of the surface.

In the PMM, the bidirectional reflectance is written as the sum of a highly directional component, ρ_h and a weakly directional component ρ_l. The reflected radiance $I_r(p,\theta_r,\phi_r)$ then can be

considered as the sum of the components $I_h(p,\theta_r,\phi_r)$ (defined by Eq.1 with ρ_h in place of ρ_{bd}) and $I_l(p,\theta_r,\phi_r)$ (defined by Eq. 1 with ρ_l in place of ρ_{bd}.) The radiance of weakly directional reflection, I_l, can be further decomposed into four components: $I_{l,s}$, the light reflected from light sources; $I_{l,c}$ the light reflected from light sources via a series of highly directional reflections, $I_{l,h}$ the light reflected from non-light sources via a series of highly directional reflections, and $I_{l,l}$ the light reflected from other weakly directional surfaces.

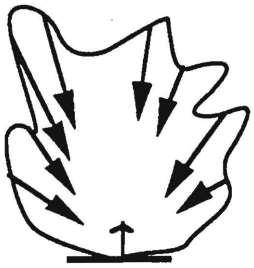
In the PMM the term $I_{l,l}$ is calculated by:

$$I_{l,l}(p,\theta_r,\phi_r) = \int \rho_l(\, p,\theta_r,\phi_r,\theta_i,\phi_i)\, I_{rad} \cos\theta_i \, d\omega_i \qquad (2)$$

where I_{rad} is the radiosity solution for the surface visible in direction i. Although this method of calculating $I_{l,l}$ is faster than doing a full path tracing solution, the computation is still extremely time consuming relative to the other illumination components. In GSII the value I_{rad} is replaced by $I_{rad,simp}$, the radiosity calculated for a geometrically simplified version of the environment.

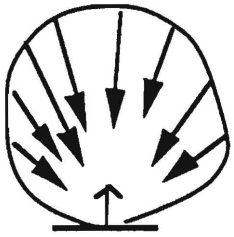
The justification for using a simplified environment for the radiosity solution is that although the simplified solution is not spatially accurate, the solution is used only to calculate weakly directional, indirect illumination. Weakly directional surfaces essentially perform an averaging function. Referring to Fig. 1, for a highly detailed spatial input (1a) the weakly directional surface reflects a blurred, or averaged distribution (1b). If the highly spatially detailed distribution is replaced by an averaged distribution (1c), there is little change in the resulting reflected distribution (1d).

The solution using GSII is shown pictorally in Fig. 2.There are two versions of the environment -- a detailed version and simplified version. The detailed version is used for all rays calculating high spatial frequency details -- view rays, specularly reflected rays, and rays to the light sources. The simplified version is used for indirect illumination rays.



(a)

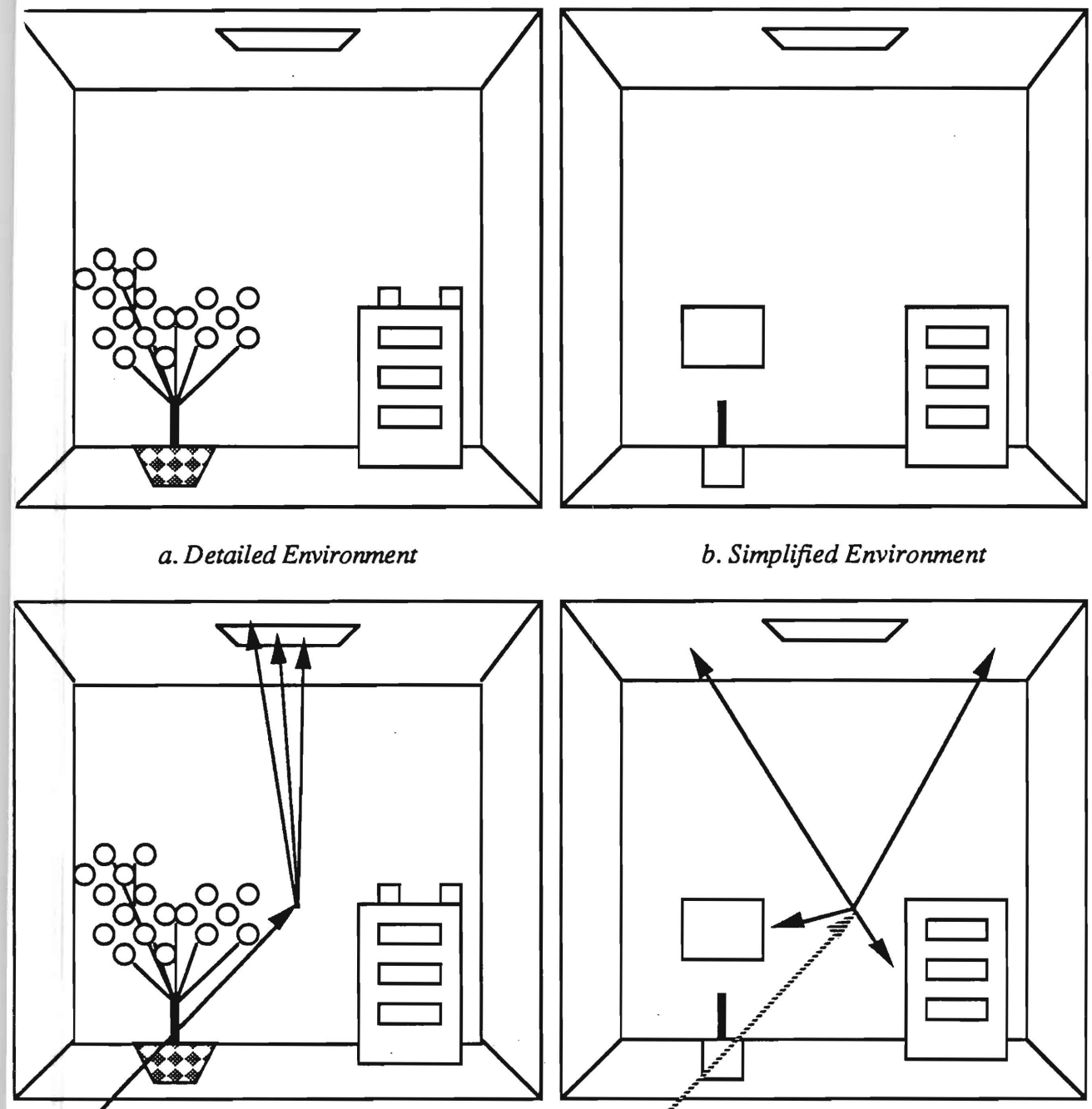
(b)



(c)

(d)

Fig. 1 A weakly directional surface with a spatially detailed incident distribution (a) reflects a blurred or averaged value (b). The same surface with a less detailed distribution with the same average energy (c) reflects approximately the same radiance distribution.



a. Detailed Environment

b. Simplified Environment

c.Rays from the eye and rays to the light source intersect the detailed environment.

d. Rays for calculating indirect illumination intersect the simplified environment

Fig. 2 - Diagram showing how detailed and simple geometric representations are used in the calculation of global illumination

3.2 Simplifying Geometries

GSII can be viewed as an extension of the patch/element hierarchy. Surfaces are simplified by methods beyond just using levels of subdivision of individual surfaces. Surfaces of a small enough size are eliminated in the simple form of the environment, and groups of large numbers of small surfaces forming objects or groups of objects are replaced by small numbers of relatively larger surfaces.

Small Isolated Surfaces The first issue we address is how small an isolated surface needs to be to be ignored in the simplified solution. To obtain a rule, we assume that the non-light source surfaces in the environment have radiances of approximately the same order of magnitude relative to the radiance of the light sources. (Reclassification of light sources in the PMM assures this.) The importance of a surface then can be estimated by the solid angle it subtends.

The solid angle subtended by the entire hemisphere above a differential surface dA is equal to 2π. The maximum solid angle $\Delta\omega$ subtended by another surface A_j is less than or equal to A_j/r^2, where r is the minimum distance from dA to a point on A_j. If $\Delta\omega$ is small compared to 2π, then A_j is small enough to ignore from the point of view of dA.

From dA, if a surface A_j is close enough, r approaches zero and surface j is too big to ignore, regardless of its absolute size. We define then a radius r_{close}, shown in Fig. 3, within which the surface di views the geometrically detailed environment, and outside of which it views a simplified environment. The maximum possible value of $\Delta\omega$ subtended by A_j then is A_j/r^2_{close}.
Let L be the characteristic length in an environment. In an indoor scene this might be the average dimension in a room. In an outdoor scene full of objects such as houses and trees, this might be the average dimension of a typical object. The radius r_{close} can be specified as a fraction of the characteristic length L, essentially indicating the level of geometric detail to be considered. The minimum area A_n for the simplified version of the environment is chosen so that $\Delta\omega_{max}$ is some fraction of 2π, essentially indicating an error level. That is:

$$r_{close} = \alpha L$$

$$A_{min}/r_{close}2 = \beta 2\pi$$

$$\Rightarrow A_{min} = 2\pi\beta\alpha^2L^2 \qquad (3)$$

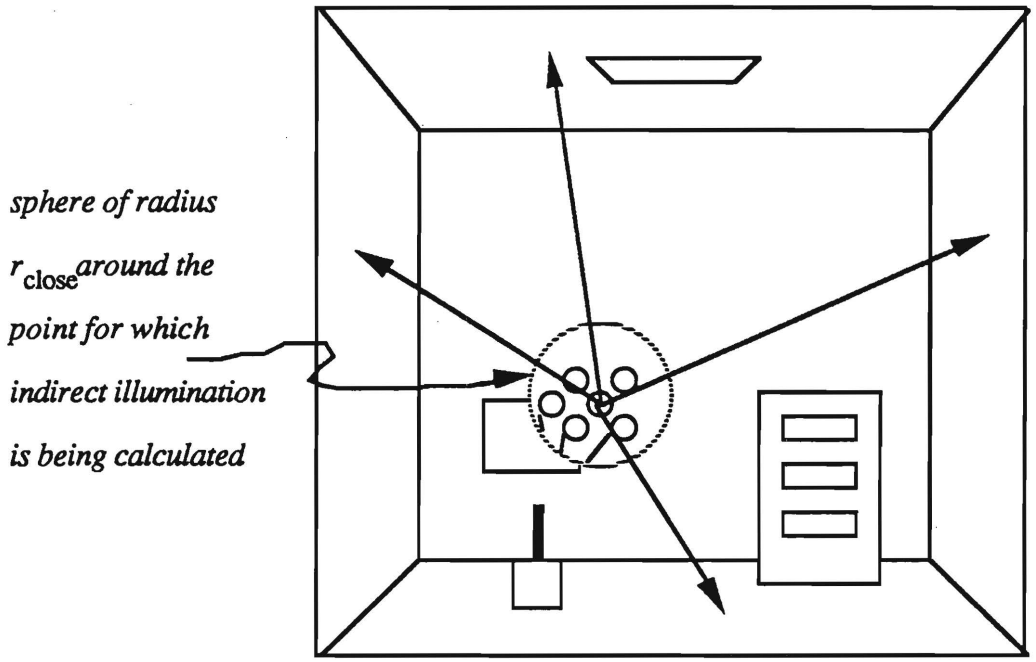


Fig. 3: Use of r_{close} in indirect illumination calculation.

Equation (3) gives the minimum area allowed in the simplified version of the environment in terms of the size of the environment, and the level of detail (α) and accuracy (β) specified by the user. For a given β, α needs to be adjusted for efficiency. The larger the value of α the larger the potential number of surface intersection tests for indirect rays, and the smaller the number of surfaces in the simplified environment.

This approach is similar in spirit to the radiosity method proposed by Xu et al. in which detailed form factors are calculated only locally. Unlike their method, however, the local area is not a fixed subdivision of space, but a region which "floats" with the particular location where illumination is being calculated. In its definition of locality, GSII is closer to the method presented by Hanrahan et al.

Clusters of Small Surfaces There are frequently groups of small surfaces which together compose a large object, such as a potted plant in a room. Such clusters cannot simply be deleted from the environment. These clusters are replaced by a simple, in some sense optically equivalent, box.

We define three types of surfaces: "normal" surfaces which are large enough to be used both in the radiosity solution and in the final rendering, "complex" surfaces which are small and are not used

until the final rendering, and "simple" surfaces which are used in place of clusters of complex surfaces in the radiosity solution.

To find appropriate "simple" surfaces, the criterion chosen for "optical equivalence" was that the replacement box should absorb the same fraction of light as the complex object. With this criterion, the proper energy balance in the environment is maintained. Two types of equivalent boxes were considered.

The first approach is diagrammed in Fig. 4. The minimum box aligned with the coordinate system which encloses the object is found. A random set of rays is sent into the box from many directions. The diffuse reflectance of each side of the box is set equal to the number of rays that are sent into the face which emerge again from the box.

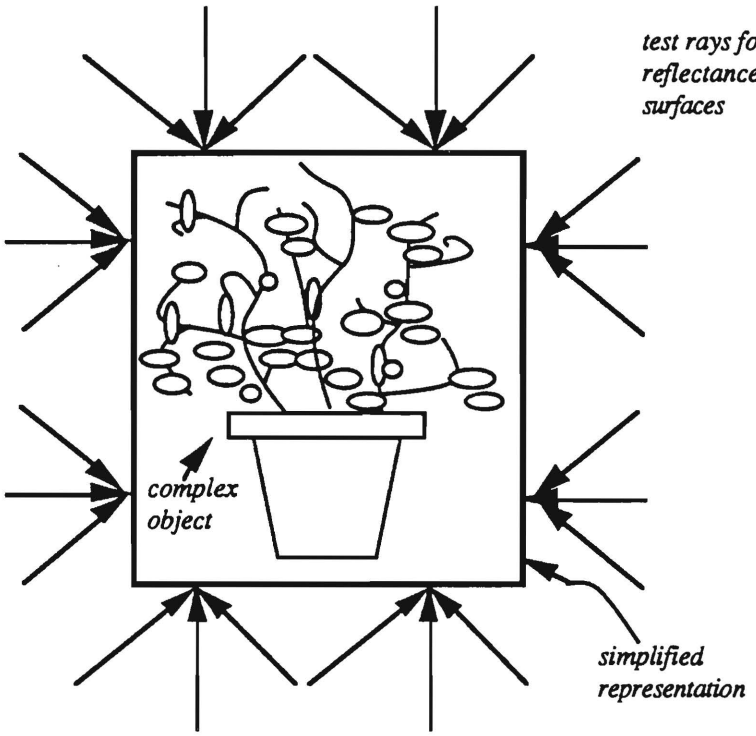


Fig. 4: One method for replacing a complex object by a simple object with reflectances determined by shooting a series of trial rays.

In pseudocode:

```
For each side of the box {
        Count = 0;
        For each point on the side 1 to P{
```

```
            For each direction 1 to D{
                    N = 1;
                    Shoot a ray into the box;
                    While in ray is still in the box{
                            N *= ρ(surface hit);
                            Follow the reflected ray;}
                    Count += N;}}}
    ρ(side) = Count/(P*D);}
```

The application of this first approach to complex objects with even moderate surface densities gave unacceptable results. Enough rays passed directly through the object to result in an excessively high reflectance. The result was an absurdly bright object appearing in the radiosity preprocess. While the average energy balance would be preserved in the final rendering, this is unacceptable for a system in which a user watches the image evolve.

The second approach was to determine the size of a smaller box. Again, a minimum box was found around the object. A set of N rays were sent at random locations through each face of the box, perpendicular to the sides of the box, as shown in Fig. 5. The size of the box was based on the fraction of the rays which hit the complex object, and the size of the minimum bounding box. The reflectance of each side of the bounding box was set equal to the average reflectance of the surfaces hit.

In pseudo-code:

```
For each side of the box sized X by Y by Z{
        Count = 0;
        rho_ave = 0;
        For N randomly located rays perpendicular to the side{
                Shoot a ray into the box;
                If a surface is hit{
                        Count += 1;
                        rho_ave += rho of surface hit}}
        rho(side) = rho_ave/count;
        frac(side) = Count/N;}
new_x_length = sqrt(frac(xy_face)frac(xz_face))*X;
new_y_length = sqrt(frac(xy_face)frac(zy_face))*Y;
new_z_length = sqrt(frac(xz_face) frac(yz_face))*Z;
```

This second approach gives simplified boxes which give a better representation of the object in the radiosity pre-process step.

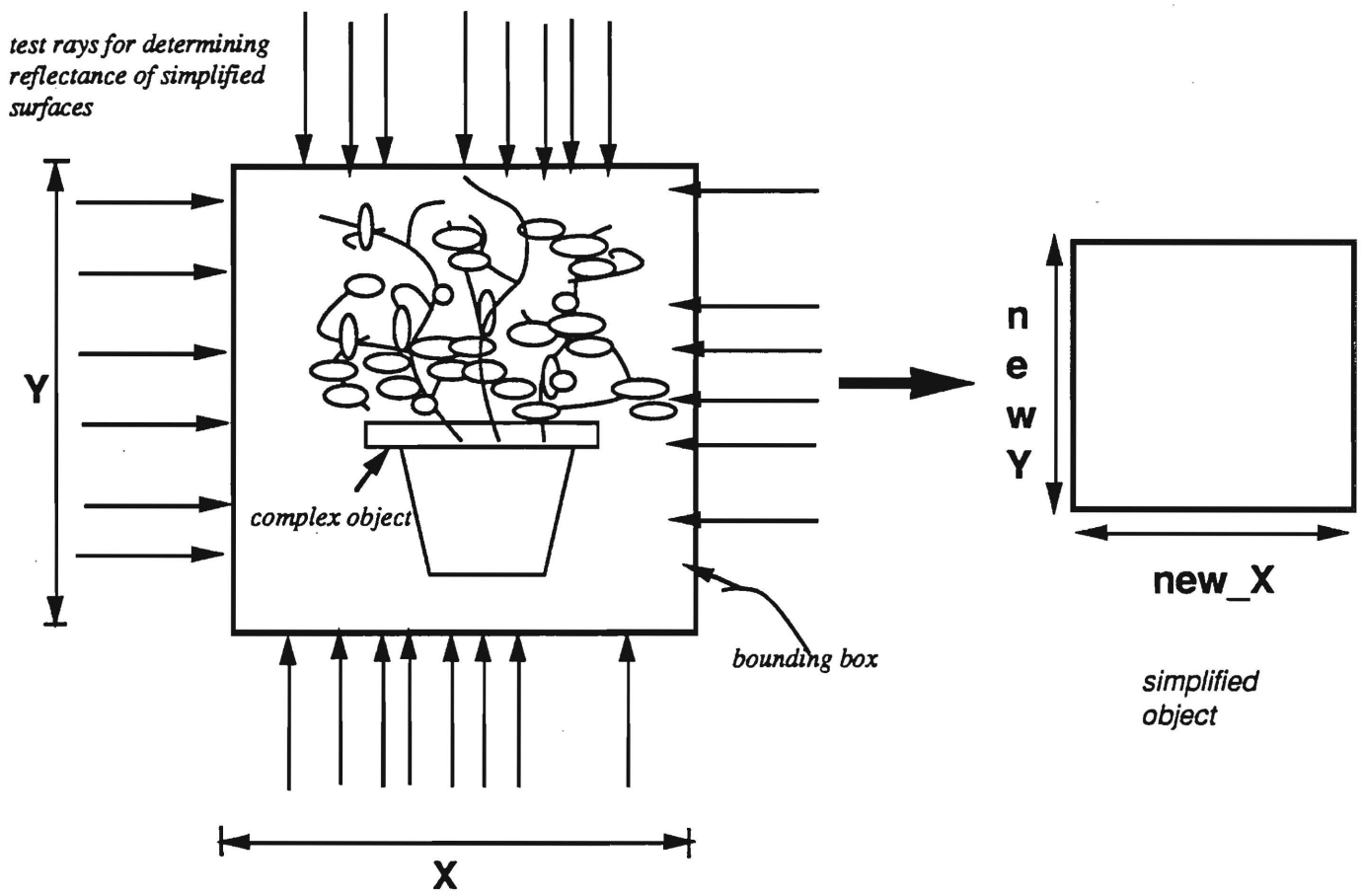


Fig. 5 An alternative method for finding the simplified object definition.

4.Implementation of a Preliminary GSII System

A preliminary rendering system using GSII is diagrammed in Fig. 6. After creation in the modeller, all surfaces are converted to the format used by the rendering programs. A separate file of objects to be simplified is also converted, and run through the Simplifier module. All of the descriptions are then merged and sent to the Radiosity module which calculates radiosities for the normal and simple surfaces. The Renderer module uses the complete geometric description and the radiosities to compute the final image.

4.1 Simplifier

In our preliminary implementation, clusters of small surfaces to be replaced are identified in the modelling process.We used the second method for simplification of clusters described in section 3.

User supplied values of α and β are used to determine whether the remaining surfaces should be classified as "normal" or "complex".

A final version of the GSII method would need an automated method for identifying such clusters. Our approach is similar to the development of meshing for radiosity. A very simple method was used initially (e.g. Goral et al. 1984). Once the utility of the radiosity method was established, more sophisticated meshing methods were developed (e.g. Baum et al. 1991).

4.2 Radiosity

The radiosity solution is calculated by a progressive refinement program which uses the hemi-cube algorithm with a hardware Z-buffer to find form factors. The solver acts only on normal and simple surfaces. Each surface is subdivided into triangular patches. The hemi-cube resolution and per cent original unshot energy stopping criterion are set by the user.

4.3 Renderer

The renderer has two passes. In both all ray casting is performed using uniform spatial subdivision to reduce ray surface intersections. The first pass is the high spatial frequency (HIRES) pass which accounts for direct illumination, specular reflections and caustics. Rays in this pass, as shown in Fig. 2, must use the normal + complex surfaces to capture the fine detail. At the end of the HIRES pass two values are stored for each pixel for each wavelength band -- the radiance I_{HIRES} , and $I_{HIRES,dev}$ the sample standard deviation . For each pixel, trial values are calculated until either the ratio ($I_{HIRES,dev}/I_{HIRES}$) falls below a user supplied value, or a user supplied maximum number of trials is reached.

The second pass is the low spatial frequency (LORES) pass. This pass accounts for the $I_{l,l}$ term in Eq. (2). In the LORES pass the rays from the first visible object out to the environment consider normal + complex surfaces only within a radius r_{close}. If a surface is intersected within this radius, the illumination calculation continues as in a Monte Carlo path tracing solution. Beyond this radius, only normal + simple surfaces are intersected, and the radiance of any surface beyond r_{close} is taken to be the radiance from the radiosity solution. The values I_{LORES} and $I_{LORES,dev}$ are calculated in this pass. Trial values are calculated until either the ratio $(I_{HIRES,dev}+I_{LORES,dev})/(I_{HIRES}+I_{LORES})$ falls below a user supplied value, or a user supplied maximum number of trials is reached. In the final image the values of $I_{HIRES} + I_{LORES}$ are displayed.

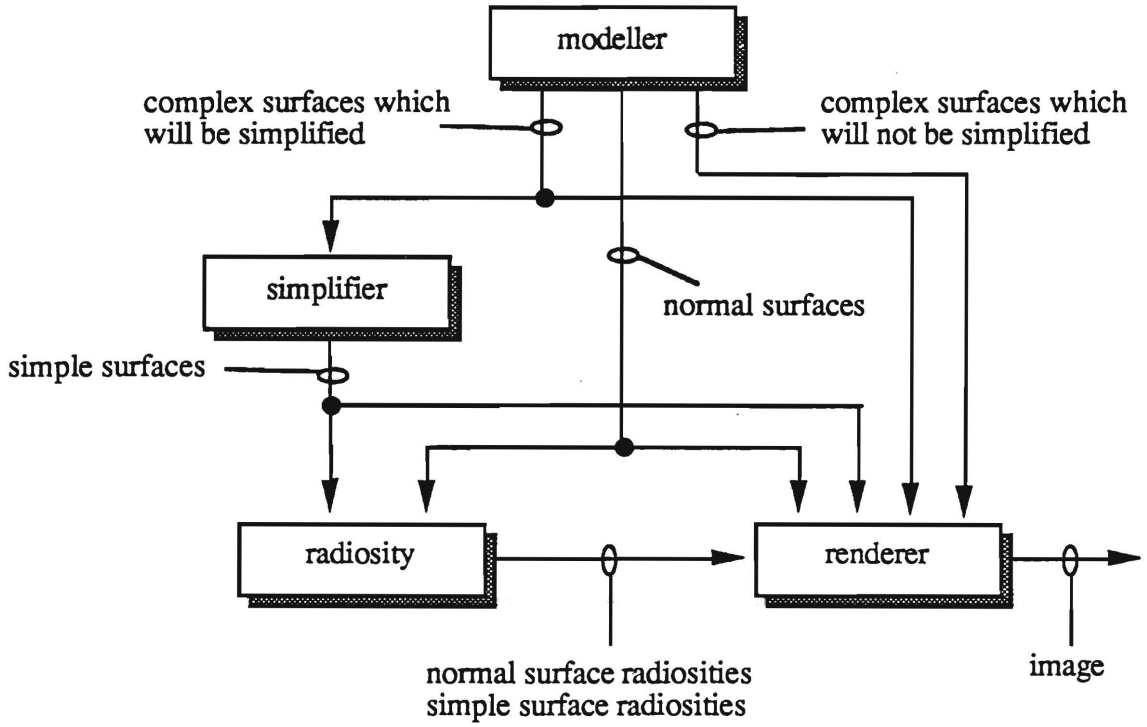


Fig. 6 Structure of preliminary GSII rendering system.

5. Results

To evaluate the effectiveness of GSII, we built a test environment composed of 10,948 surfaces. These surfaces made up the objects in an office including thumb tacks, pens, a computer keyboard and a plant. (See slide 1 for close-ups of these objects.) We considered two cases -- one in which surfaces in the environment have relatively low reflectances ($\rho_{ave} = 0.41$), and indirect illumination is relatively unimportant, and the other in which surfaces have relatively high reflectances ($\rho_{ave} = 0.56$) and indirect illumination is more important. (The adjustments in average reflectance were made by adjusting the wall reflectances only.) To make comparisons we generated images for each case using progressive refinement radiosity (PRR) with normal + complex surfaces, PRR with normal + simple surfaces, Monte Carlo path tracing (MCPT), PMM, and with the new GSII.

In solving for the original and simplified scene with PRR, we used a hemi-cube of 150x150 pixels per full face. The normal + complex environment contained 10,948 surfaces divided into 74,924 patches. We did not use element or sub-element subdivision. The normal + simple environment contained 105 surfaces subdivided into 1124 patches. We continued the solution until one per cent of the original energy in the room was left unshot. The timings for the PRR solutions are

summarized in Table 1. Images showing the PRR results (without smoothing) are the two top images of slide 2 for the lower reflectance room, and the two top images of slide 4 for the higher reflectance room. In each case, the image on the left is using the simplified database and the image on the right is using the original database.

Table 1.-- Timings* for PRR Preprocess

Average Reflectance	Representation	# Patches	Timing (hrs:min)
Low (0.41)	normal + simple	1124	0:25
Low (0.41)	normal + complex	74924	50:25
High (0.56)	normal + simple	1124	0:52
High (0.56)	normal + complex	74924	**142:14

*Timings for SGI 4D/20 Personal Iris
**Solution did not run to completion

The simplified geometry contains far fewer surfaces than the original, and the result is orders of magnitude lower timings for the PRR on the simplified geometry. While the absolute values of the timings are machine and implementation specific, the ratio of timings for the simple and complex solutions are significant.

A 12x12x12 uniform spatial subdivision was used for all ray casting in the rendering phase. Solutions for each pass were allowed to run until the sample standard deviation was equal to 5 per cent of the computed value, or a maximum of 30 trials was reached. A minimum of 10 trials were run for each pixel. Table 2 shows the timing results for the HIRES pass for each case. The two lower images in slide 2 show the results for the low reflectance room, and the two lower images in slide 4 for the high reflectance room. The image on the left in each case shows the values of $I_{HIRES,dev}$ (displayed with a very high scaling factor to make the values visible), and the image on the right shows the values of I_{HIRES}.

Table 2.-- Timings* for HIRES Pass

Average Reflectance	Average I_{dev}/I	Timing (hrs:min)
Low (0.41)	0.074	3:17
High (0.56)	0.074	3:16

*Timings for SGI 4D/280 using only 33 MHz processors

Table 3 shows the time results for each case for each method for the LORES pass. The advantage of using either the PMM or GSII over the MCPT method is clear when the timings are compared. Furthermore, the increase in time for solution for the MCPT with the increased average reflectance is evident, while the timings for the PMM and GSII are the same for both the low and high cases.

Table 3.-- Timings* for LORES Pass

Average Reflectance	Method	Average I_{dev}/I	Timing (hrs:min)
Low (0.41)	MCPT	0.349	7:22
Low (0.41)	PMM	0.280	3:39
Low (0.41)	GSII	0.276	4:14
High (0.56)	MCPT	0.282	10.29
High (0.41)	PMM	0.189	3:47
High (0.41)	GSII	0.193	4:17

*Timings for SGI 4D/280 using only 33 MHz processors

The comparison of PMM and GSII timings show that they are approximately the same. In this instance the use of spatial subdivision equalized the cost of casting rays into the simple and complex environments. Also, it appears that the variance in environment illumination was not significantly different between the complex and simple radiosity solutions.

The images formed by the MCPT, PMM, and GSII are shown in slides 3 and 5 for the low and high reflectance cases respectively (MCPT on top, PMM on bottom, left, GSII on bottom, right). The major differences in each case are some artifacts in the PMM images on the back wall. These result from taking the radiosity of surfaces which are too close. That is, although the radiosity solution is correct on average, some individual surfaces are too bright and some too dark. In the PMM errors are produced when a surface uses the erroneous radiosity of a nearby surface which subtends a large solid angle. By using the radius r_{close} in the GSII these types of errors are avoided.

These preliminary results suggest two major conclusions -- 1.) GSII can be used in place of MCPT or PMM to generate images without a degradation in the results and 2.) the major time savings in using GSII over the PMM is in greatly reducing the time to perform the radiosity preprocess. Many more tests are required to confirm these conclusions.

6. Summary and Future Work

We have described and presented a preliminary implementation of a global illumination method that uses simplified geometric representations for indirect illumination. We have presented initial results which indicate that the method can produce images more quickly than Monte Carlo path tracing or the progressive muti-pass method without degradation of quality.

Potential future work is in three areas -- testing, simplification methods, and animation. Clearly more tests are required to examine the relative timings for various types of environments. Detailed analyses of the accuracy of the results of the various methods should also be made. Automated geometric simplification methods are needed if the approach is to become practical. Finally, the use of spatially and temporally less detailed global illumination solutions as a preprocess for animated sequences could be explored.

7. Acknowledgements

This work was supported by grants from the National Science Foundation and the Apple Computer Company.

8. References

Baum, D.R., S. Mann, K.P. Smith, and J.M.Winget, 1991, "Making Radiosity Usable:Automatic Preprocessing and Meshing Techniques for the Generation of Accurate Radiosity Solutions," *Computer Graphics* (Proceedings of SIGGRAPH 1991) 25(4), 51-60.

Chen, S. E., H. E. Rushmeier, G. Miller and D. Turner, 1991,"A Progressive Multi-Pass Method for Global Illumination," *Computer Graphics* (Proceedings of SIGGRAPH 1991) 25(4), 165-174.

Cohen, M.F., D.P. Greenberg, D.S. Immel and P.J. Brock, 1986, "An Efficient Radiosity Approach for Realistic Image Synthesis," *IEEE Computer Graphics and Applications*, 6(2), 26-35.

Cohen, M.F., S.E. Chen, J.R. Wallace, and D.P. Greenberg, 1988, "A Progressive Refinement Approach to Fast Radiosity Image Generation," *Computer Graphics* (Proceedings of SIGGRAPH 1988) 22(4), 75-84.

Goral, C.M., K.E. Torrance, D.P. Greenberg and B. Battaille, 1984, "Modelling Interreflections Between Diffuse Surfaces," *Computer Graphics* (Proceedings of SIGGRAPH 1984) 18(3), 213-222.

Hanrahan, P., D. Salzman, and L. Aupperle, 1991, " A Rapid Hierarchical Radiosity Algorithm," *Computer Graphics* (Proceedings of SIGGRAPH 1991) 25(4), 197-206.

Kajiya, J., 1985, "Anisotropic Reflectance Models," *Computer Graphics* (Proceedings of SIGGRAPH 1985) 19(3), 15-21.

Kajiya, J., 1986, "The Rendering Equation," *Computer Graphics* (Proceedings of SIGGRAPH 1986) 20(4), 143-150.

Nishita, T. and E. Nakamae, 1985, "Continuous Tone Representation of Three-Dimensional Objects Taking Account of Shadows and Interreflection," *Computer Graphics* (Proceedings of SIGGRAPH 1985) 19(3), 23-30.

Xu, H. P., Q.-S. Peng, and Y.-D. Liang, 1990, "Accelerated Radiosity Method for Complex Environments," *Computers and Graphics*, 14(1), 65-71.